PATENT
P56992

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

BYOUNG-CHUL KIM *et al.*

Serial No.: *to be assigned* Examiner: *to be assigned*

Filed: 1 December 2003 Art Unit: *to be assigned*

For: DYNAMIC MANAGEMENT METHOD FOR FORWARDING INFORMATION IN ROUTER HAVING DISTRIBUTED ARCHITECTURE

CLAIM OF PRIORITY
UNDER 35 U.S.C. §119

Mail Stop Patent Application
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

The benefit of the filing date of the following prior foreign application, Korean Priority No. 2002-75701 filed in Korea on 30 November 2002, and filed in the U.S. Patent and Trademark Office on 1 December 2003, is hereby requested and the right of priority provided in 35 U.S.C. §119 is hereby claimed.

In support of this claim, filed herewith is a certified copy of said original foreign application.

Respectfully submitted,

Robert E. Bushnell
Reg. No.: 27,774
Attorney for the Applicant

Suite 300, 1522 "K" Street, N.W.
Washington, D.C. 20005-1202
(202) 408-9040
Folio: P56992
Date: 1 December 2003
I.D.: REB/wc

대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 10-2002-0075701
Application Number PATENT-2002-0075701

출 원 년 월 일 : 2002년 11월 30일
Date of Application NOV 30, 2002

출 원 인 : 삼성전자 주식회사
Applicant(s) SAMSUNG ELECTRONICS CO., LTD.

2003 년 01 월 16 일

특 허 청
COMMISSIONER

【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【참조번호】	0003
【제출일자】	2002.11.30
【국제특허분류】	H04L
【발명의 명칭】	분산구조라우터에서 포워딩 정보를 동적으로 관리하는 방법
【발명의 영문명칭】	DYNAMIC MANAGEMENT METHOD FOR FORWARDING INFORMATION IN DISTRIBUTED ROUTER ARCHITECTURE
【출원인】	
【명칭】	삼성전자 주식회사
【출원인코드】	1-1998-104271-3
【대리인】	
【성명】	이건주
【대리인코드】	9-1998-000339-8
【포괄위임등록번호】	1999-006038-0
【발명자】	
【성명의 국문표기】	김병철
【성명의 영문표기】	KIM,BYOUNG CHUL
【주민등록번호】	711025-1067017
【우편번호】	449-753
【주소】	경기도 용인시 수지읍 동성1차아파트 101동 103호
【국적】	KR
【발명자】	
【성명의 국문표기】	문세웅
【성명의 영문표기】	MOON,SE WOONG
【주민등록번호】	671104-1029818
【우편번호】	463-906
【주소】	경기도 성남시 분당구 이매동 한신아파트 209동 1302호
【국적】	KR

1020020075701 출력 일자: 2003/1/18

〖발명자〗

〖성명의 국문표기〗 최병구

〖성명의 영문표기〗 CHOE,BYUNG GU

〖주민등록번호〗 570226-1140316

〖우편번호〗 150-834

〖주소〗 서울특별시 영등포구 문래동3가 54번지 문래엘지빌리지 117동 1501호

〖국적〗 KR

〖취지〗 특허법 제42조의 규정에 의하여 위와 같이 출원합니다. 대리인 주 (인) 이건

〖수수료〗

〖기본출원료〗	20	면	29,000 원
〖가산출원료〗	34	면	34,000 원
〖우선권주장료〗	0	건	0 원
〖심사청구료〗	0	항	0 원
〖합계〗	63,000	원	

【요약서】

【요약】

본 발명은 라우팅 노드들을 다수개 포함하는 분산구조라우터에서 포워딩정보를 관리하는 방법에 관한 것으로서, 본 발명은 포워딩정보의 추가 또는 삭제에 응답하여 포워딩정보를 동적으로 어그리게이션(aggregation) 또는 디스어그리게이션(disaggregation)함으로써, 분산구조라우터의 각 라우팅노드들에서 관리하는 포워딩테이블의 크기를 줄일 수 있다. 또한, 본 발명은 분산구조라우터에서 포워딩테이블을 업데이트하기 위해 전송되는 제어-패킷(control packet)의 전송량을 줄여 내부 트래픽을 줄일 수 있다는 효과가 있다.

【대표도】

도 5

【색인어】

분산구조라우터, 포워딩정보관리, 어그리게이션

【명세서】

【발명의 명칭】

분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법{DYNAMIC MANAGEMENT METHOD FOR FORWARDING INFORMATION IN DISTRIBUTED ROUTER ARCHITECTURE}

【도면의 간단한 설명】

도 1은 분산구조라우터에 대한 예시도,

도 2는 분산구조라우터의 통상적인 프로세스 구조를 도시한 도면,

도 3은 본 발명의 일 실시 예에 따라 포워딩정보를 관리하기 위한 분산구조라우터의 프로세스 구조를 도시한 도면,

도 4a는 본 발명의 일 실시 예에 따라 포워딩정보를 관리하기 위해 생성된 어그리게이션 트리(aggregation tree)의 각 노드별 관리데이터 구조에 대한 예시도,

도 4b는 본 발명의 일 실시 예에 따라 포워딩정보를 관리하기 위해 생성된 어그리게이션 트리(aggregation tree) 구조에 대한 예시도,

도 4c는 포워딩주소정보(prefix)의 길이별 분포를 나타낸 그래프,

도 5는 새롭게 추가된 포워딩정보를 본 발명의 일 실시 예에 따라 관리하는 방법에 대한 처리 흐름도,

도 6은 본 발명의 일 실시 예에 따라 로컬영역정보를 추가하는 과정에 대한 처리 흐름도,

도 7은 본 발명의 일 실시 예에 따라 가상영역정보를 추가하는 과정에 대한 처리 흐름도,

도 8은 삭제된 포워딩정보를 본 발명의 일 실시 예에 따라 관리하는 방법에 대한 처리 흐름도,

도 9는 본 발명의 일 실시 예에 따라 로컬영역정보를 삭제하는 과정에 대한 처리 흐름도,

도 10은 본 발명의 일 실시 예에 따라 가상영역정보를 삭제하는 과정에 대한 처리 흐름도,

도 11a 및 도 11b는 새롭게 추가된 포워딩정보를 관리하는 방법을 도식화하여 설명한 도면,

도 12a 및 도 12b는 삭제된 포워딩정보를 관리하는 방법을 도식화하여 설명한 도면,

도 13a 내지 도 13d는 본 발명의 일 실시 예에 따른 포워딩정보 동적 관리방법의 효과를 입증하기 위한 테스트 결과들,

도 14a 내지 도 14j는 본 발명의 일 실시 예에 따른 포워딩정보를 동적으로 관리하는 방법에 대한 알고리즘 예를 나타낸 도면.

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<17> 본 발명은 분산구조라우터에서 포워딩정보를 관리하는 방법에 관한 것으로서, 특히, 분산구조라우터에서 생성되는 포워딩정보를 어그리게이션(aggregation) 또는 디스어그리게이션(disaggregation)에 의해 동적으로 관리하는 방법에 관한 것이다.

<18> 대용량 초고속 네트워크의 발전에 따라 라우터는 기존의 중앙집중형 구조에서 분산처리형의 새로운 구조로 변화해가고 있다.

<19> 중앙집중형 라우터는 중앙의 프로세서에서 라우팅 프로토콜을 구현하고, 그 라우팅 프로토콜이 수집한 라우팅정보들을 상기 중앙의 프로세서가 관리하였다. 예컨대, 중앙집중형 라우터에서는 중앙의 프로세서가 라우팅 테이블을 계산하고, 각각의 라인카드에 라우팅 테이블을 분배해주는 역할을 하였다. 따라서, 라인 카드에 의한 패킷 포워딩이 중앙의 라우트 프로세서로부터 전해지는 라우팅 테이블 정보에 의해서 이루어졌다.

<20> 반면, 분산처리형 라우터는 이와 같이 중앙의 프로세서에 집중된 작업들을 다수개의 프로세서에 의해 분산처리 하도록 한다. 따라서, 분산처리형 라우터는 중앙집중형 라우터에 비해 대용량의 데이터를 처리할 수 있다. 예를 들어, 분산처리형 라우터에서는 라우팅 프로토콜을 관리하는 프로세서와 라우팅테이블을 계산하는 프로세서와 패킷 포워딩을 관리하는 프로세서들을 각각 따로 두고, 각각의 프로세서에서 해당 작업들을 분산처리함으로써 라우팅 성능을 향상시켰다.

<21> 도 1은 분산구조라우터에 대한 예시도이다. 특히, 도 1은 갤럭시 아이.피. 라우터(Galaxy IP Router)의 예를 나타내었다.

<22> 도 1을 참조하면, 분산구조라우터(100)는 다수개의 라우팅 노드들(RNs: Routing Nodes)(110, 120, 130, 140)을 포함하고, 그 RNs(110, 120, 130, 140)은 스위칭 모듈(SWM: switching module)(150)에 의해 서로 연결된다. 한편, 각각의 RN(110, 120, 130, 140 중 어느 하나)에는 입출력 프로세서(IOP: Input Output Processor)(111)가 탑재되고, 그 IOP(111)는 두 개의 물리적인 매체들(PMD-A(Physical Medium Device A)(112) 또는 PMD-B(113))로부터 패킷을 전달받도록 설계되었다(designed).

<23> 이러한 RNs(110, 120, 130, 140)은 각각 대응되는 서브-넷(sub-network)을 지원하기 위한 고유의 라우팅 테이블(own routing table)과 그 라우팅 경로를 처리하기 위한 고유의 프로세서들(own processors)을 가진다. 그리고, 모든 RNs(110, 120, 130, 140)은 이를 통해 독자적인 라우팅 프로토콜을 실행시키고(running), 독자적으로 포워딩(fowarding)을 수행한다. 하지만, 이러한 RNs(110, 120, 130, 140)은 사용자의 관점에서는 하나의 라우터로 인식되어진다. 이를 위해, RNs(110, 120, 130, 140)은 SWM(150)을 통해 연결된 다른 RNs(110, 120, 130, 140)의 라우팅 테이블들을 통합적으로(globally) 관리한다. 이 때, 각 RN(110, 120, 130, 140 중 어느 하나)에 실제로 연결된 물리적인 서브-넷(sub-network)을 로컬영역(Local area)(B)이라 하고, SWM(150)을 통해 RNs(110, 120, 130, 140)이 연결되어 형성된 네트워크를 가상영역(Virtual area)(A)이라 한다.

<24> 도 2는 분산구조라우터의 통상적인 프로세스 구조를 도시한 도면이다. 특히, 도 2에서는 분산구조라우터(100)가 4개의 RNs을 갖는 경우에 각 RN에 탑재된 IOP의 프로세스 구조 및 그 IOP들(IOP#1(10), IOP#2(20), IOP#3(30), IOP#4(40))이 SWM(50)을 통해 연결된 상태를 도시하였다.

<25> 도 2를 참조하면, IOP#1(10)은 다수개의 라우팅 프로토콜들(ripd(11), ospfd(12), bgpd(13), isisd(14))과, IOP 관리 프로세서(glued: Galaxy Loosely Unified Environment Daemon)(15), 라우팅테이블(routing table)(16), 포워딩테이블(forwarding table)(17)을 포함한다.

<26> 라우팅 프로토콜들(ripd(11), ospfd(12), bgpd(13), isisd(14))은 각각 고유의 수집기준에 의해 라우팅 정보들을 수집한다. 라우팅테이블(16)은 라우팅 프로토콜들(ripd(11), ospfd(12), bgpd(13), isisd(14))에서 수집된 라우팅 정보들을 저장한다. 포워딩테이블(17)은 라우팅테이블(16)에 저장된 라우팅 정보들을 계산하여 얻어진 포워딩정보를 저장한다.

<27> IOP 관리 프로세서(glued)(15)는 라우팅 프로토콜들(ripd(11), ospfd(12), bgpd(13), isisd(14))에서 수집된 라우팅 정보들을 라우팅테이블(16)에 저장하고, 그 라우팅 정보들을 계산하여 얻어진 포워딩정보를 포워딩테이블(17)에 저장하는 일련의 과정들을 수행하고 관리한다. 또한, IOP 관리 프로세서(glued)(15)는 이 때 얻어진 라우팅 테이블(16)을 SWM(50)을 통해 다른 IOP들(IOP#2(20), IOP#3(30), IOP#4(40))에게 통지(advertise)한다.

<28> 한편, IOP#1(10)은 라우팅 프로토콜들(ripd(11), ospfd(12), bgpd(13), isisd(14)), IOP 관리 프로세서(glued)(15) 및 라우팅테이블(16)을 포함하는 시스템 프

로세서(system processor)영역과, 포워딩테이블(17)을 포함하는 네트워크 프로세서(network processor)영역으로 구분된다. 시스템 프로세서(system processor) 영역은 라우팅정보의 수집, 라우팅경로 계산에 의한 포워딩테이블 관리 및 타 IOP들(IOP#2(20), IOP#3(30), IOP#4(40))과의 라우팅테이블 공유를 위한 일련의 처리 과정들을 수행하고, 네트워크 프로세서(network processor)는 포워딩테이블(17)에 의거하여 로컬영역(local area)에 연결된 네트웍장비들간의 포워딩을 수행한다. 이로 인해, 분산구조라우터가 대용량의 데이터를 보다 신속하게 처리할 수 있게 되는 것이다.

<29> 도 1 및 도 2에 예시된 것처럼 분산구조라우터(100)에서 대용량의 데이터를 보다 신속하게 처리하게 하기 위해서는 분산구조라우터(100)에 포함된 각 RN에서 개별 관리하는 포워딩테이블(forwarding table)들을 다른 RNs에서도 알 수 있도록 하는 것이 필수적이다. 이를 위해, 종래의 분산구조라우터(100)에서는 각 RN에서 개별 관리하는 포워딩테이블들을 SWM(150)을 통해 상호 전송함으로써 각 RN에서 모든 RNs의 포워딩테이블을 통합적으로(globally) 관리하도록 하고 있다. 예를 들어, 분산구조라우터(100)에 10개의 RNs이 있고 그 각각의 RN이 고유한 10,000개의 포워딩정보들(forwarding entries)을 가진다면, 각 RN은 100,000(=10,000*10(RNs))의 포워딩정보들을 관리하여야 한다. 따라서, 종래의 분산구조라우터(100)들은 이러한 포워딩테이블을 저장하기 위한 대용량의 저장공간을 필요로 했으며, 패킷 포워딩에 오버헤드가 발생한다는 문제점이 있었다.

【발명이 이루고자 하는 기술적 과제】

<30> 본 발명은 상기와 같은 문제점을 해결하기 위해 안출된 것으로서, 본 발명의 제1 목적은 분산구조라우터의 각 RN에서 관리하는 포워딩테이블의 크기를 줄이기 위한 방법을 제공하는 데에 있다.

<31> 본 발명의 제2 목적은 분산구조라우터에서 포워딩테이블을 업데이트하기 위해 전송되는 제어-패킷(control packet)의 전송량을 줄여 내부 트래픽을 줄이기 위한 방법을 제공하는 데에 있다.

<32> 본 발명의 제3 목적은 분산구조라우터에서 라우팅정보의 추가 또는 삭제에 응답하여 포워딩정보를 어그리게이션(aggregation) 또는 디스어그리게이션(disaggregation)에 의해 동적으로 관리하는 방법을 제공하는 데에 있다.

【발명의 구성 및 작용】

<33> 상기 목적을 달성하기 위해 본 발명에서 제공하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법은 라우팅 노드들을 다수개 포함하는 분산구조라우터에서 포워딩정보를 관리하는 방법에 있어서, 임의의 한 라우팅 노드에 새로운 포워딩정보가 삽입되면, 그 포워딩정보에 의거하여 어그리게이션 트리를 구성 및 변경하는 제1 과정과, 상기 제1 과정에서 삽입된 포워딩정보가 상기 라우팅 노드의 로컬 영역에서 생성되었는지의 여부를 판단하는 제2 과정과, 상기 제2 과정의 판단결과 상기 포워딩정보가 상기 라우팅 노드의 로컬 영역에서 생성된 경우, 상기 어그리게이션 트리를 분석하여 상기 포워딩정보를 다른 라우팅 노드들에게 통지(advertise)할 지의 여부를 판단하는 제3 과정

과, 상기 제3 과정의 판단결과에 의거하여 상기 포워딩정보를 다른 라우팅 노드들에게 통지(advertise)한 후 그 포워딩정보를 해당 라우팅 노드의 로컬 포워딩테이블에 저장하는 제4 과정과, 상기 제2 과정의 판단결과 상기 포워딩정보가 상기 라우팅 노드의 로컬 영역에서 생성되지 않은 경우, 상기 어그리게이션 트리를 분석하여 상기 포워딩정보를 해당 라우팅 노드의 로컬 포워딩테이블에 저장할 지의 여부를 판단하는 제5 과정과, 상기 제5 과정의 판단결과에 의거하여 상기 포워딩정보를 해당 라우팅 노드의 로컬 포워딩테이블에 저장하는 제6 과정을 포함하는 것을 특징으로 한다.

<34> 또한, 상기 목적을 달성하기 위해 본 발명에서 제공하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법은 라우팅 노드들을 다수개 포함하는 분산구조라우터에서 포워딩정보를 관리하는 방법에 있어서, 임의의 한 라우팅 노드에서 포워딩정보가 삭제되면, 그 라우팅 노드 내에 기 설정된 어그리게이션 트리로부터 그 포워딩정보에 해당되는 노드정보를 추출하여 상기 포워딩정보가 상기 라우팅 노드의 로컬 영역에서 생성되었는지의 여부를 판단하는 제1 과정과, 상기 제1 과정의 판단결과 상기 포워딩정보가 상기 라우팅 노드의 로컬 영역에서 생성되고 그 포워딩정보가 다른 라우팅 노드들에게 통지된 경우에만 상기 포워딩정보의 삭제정보를 다른 라우팅 노드들에게 통지한 후 그 삭제된 포워딩정보를 상기 어그리게이션 트리 및 해당 라우팅 노드의 로컬 포워딩테이블에서 삭제하는 제2 과정과, 상기 제1 과정의 판단결과 상기 포워딩정보가 상기 라우팅 노드의 로컬 영역에서 생성되지 않은 경우 그 포워딩정보에 해당되는 노드를 상기 어그리게이션 트리에서 삭제하는 제3 과정을 포함하는 것을 특징으로 한다.

<35> 이하, 본 발명의 바람직한 실시 예들을 첨부한 도면을 참조하여 상세히 설명한다. 도면들 중 동일한 구성요소들은 가능한 한 어느 곳에서든지 동일한 부호들로 나타내고

있음에 유의해야 한다. 또한 본 발명의 요지를 불필요하게 흐릴 수 있는 공지 기능 및 구성에 대한 상세한 설명은 생략한다.

<36> 도 3은 본 발명의 일 실시 예에 따라 포워딩정보를 관리하기 위한 분산구조라우터의 프로세스 구조를 도시한 도면이다. 도 3을 참조하면, 본 발명의 일 실시 예에 따라 포워딩정보를 관리하기 위한 분산구조라우터의 프로세스 구조는 IOP#1(210)의 시스템 영역에 어그리게이션 트리(aggregation tree)(218)를 더 포함한다. 일반적으로 어그리게이션(aggregation)이란, 구성 또는 조직(composition)의 부분들(parts)을 캡슐화(encapsulation)한다는 의미를 가지는데, 이를 위해 어그리게이션 트리(aggregation tree)(218)에는 각 IOP들(IOP#1(210), IOP#2(220), IOP#3(230), IOP#4(240))이 관리하는 포워딩정보들에 대응되는 노드와, 그 포워딩정보들을 어그리게이션(aggregation)하는 가상노드(virtual node)가 포함된다.

<37> 예컨대, IOP#1(210)에 P1=0101101과 P2=0101100의 2개의 포워딩정보를 가지고 있는 경우, IOP#1(210)을 제외한 다른 IOP들(IOP#2(220), IOP#3(230), IOP#4(240))은 P1 및 P2의 대표격인 가상의 P3=010110에 대한 포워딩정보만 가지고도 P1 및 P2의 포워딩정보를 얻을 수가 있다. 따라서, 다른 IOP들(IOP#2(220), IOP#3(230), IOP#4(240))은 상기 P1 및 P2의 포워딩정보들을 모두 관리할 필요가 없고, P1과 P2의 대표격인 가상의 P3=010110에 대한 포워딩정보만을 관리하면 된다. 그러므로, IOP#1(210)는 어그리게이션 트리(aggregation tree)에 P1과 P2에 대한 포워딩정보에 대응되는 노드들과, 그 P1과 P2를 어그리게이션(aggregation)하는 P3에 대응되는 가상노드(virtual node)를 추가한다. 그리고, IOP#1(210)는 P1 및 P2의 포워딩정보 대신에 P3의 포워딩정보만을 다른 IOP들(IOP#2(220), IOP#3(230), IOP#4(240))에게 통지(advertise)한다.

<38> 한편, IOP#1(210)가 P1=0101101 하나의 포워딩정보를 가지고 있고, P1을 다른 IOP들(IOP#2(220), IOP#3(230), IOP#4(240))에게 통지(advertise)한 경우 다른 IOP들(IOP#2(220), IOP#3(230), IOP#4(240))은 자신의 어그리게이션 트리(aggregation tree)에 그 P1 및 상기 P3=010110에 대응되는 가상노드(virtual node)를 추가한 후, P3의 포워딩정보를 그 IOP들(IOP#2(220), IOP#3(230), IOP#4(240)) 각각의 로컬포워딩테이블(local forwarding table)에 추가한다. 그러면, 추후에 IOP#1(210)에 포워딩정보 P2=0101100가 추가되더라도, IOP들(IOP#2(220), IOP#3(230), IOP#4(240))은 그 포워딩정보 P2를 각각의 어그리게이션트리(aggregation tree) 및 로컬포워딩테이블(local forwarding table)에 추가하지 않아도 된다. 이는 기 저장된 P3에 의해 P2의 포워딩정보를 찾을 수 있기 때문이다.

<39> 하지만, 이와 같이 P1이 IOP#1(210)에 추가되어 IOP#1(210)이 P1의 포워딩정보를 IOP들(IOP#2(220), IOP#3(230), IOP#4(240))에게 통지한(advertise) 후에 P2가 추가되고, 그 후에 P1이 IOP#1(210)에서 삭제된 경우, IOP#1(210)은 P1에 대한 삭제정보를 IOP들(IOP#2(220), IOP#3(230), IOP#4(240))에게 통지(advertise)한다. 그런데, 이러한 삭제정보에 응답하여 각 IOP들(IOP#2(220), IOP#3(230), IOP#4(240))이 각각의 어그리게이션 트리(aggregation tree)에 저장된 P1 및 P3를 단순 삭제한다면, 이들(IOP#2(220), IOP#3(230), IOP#4(240))은 P2에 대한 포워딩정보를 잃어버리게 된다. 따라서, 이를 방지하기 위해, IOP#1(210)은 P1에 대한 삭제정보를 IOP들(IOP#2(220), IOP#3(230), IOP#4(240))에게 통지(advertise)한 후, P2에 대한 포워딩정보를 IOP들(IOP#2(220), IOP#3(230), IOP#4(240))에게 통지(advertise)해 준다. 그러면, P2에 대한 포워딩정보를 수신한 IOP들(IOP#2(220), IOP#3(230), IOP#4(240))은 P2를 어그리게이션

트리(aggregation tree)에 추가하고 P2의 상위에 가상노드인 P3을 다시 생성하여 어그리게이션 트리(aggregation tree) 및 로컬포워딩테이블(local forwarding table)에 추가한다. 이를 디스어그리게이션(disaggregation)이라 한다.

<40> 어그리게이션 트리(aggregation tree)(218)는 이러한 어그리게이션(aggregation) 또는 디스어그리게이션(disaggregation) 수행을 위해 참조되는 것이다. 한편, 도 3에 예시된 그 밖의 구성들은 도 2에 예시된 바와 유사하므로, 구체적인 설명은 생략하였다.

<41> 도 4a는 본 발명의 일 실시 예에 따라 포워딩정보를 관리하기 위해 생성된 어그리게이션 트리(aggregation tree)의 각 노드별 관리데이터 구조에 대한 예시도이다. 도 4a를 참조하면, 본 발명의 일 실시 예에 따라 포워딩정보를 관리하기 위한 어그리게이션 트리(aggregation tree)의 각 노드들은 해당 포워딩정보가 포워딩될 주소정보(Prefix), 그 포워딩될 주소의 길이정보(Length), 포워딩정보의 타입정보(Type), 포워딩정보를 생성한 라우팅노드 정보(Source IOP), 해당 포워딩정보를 다른 라우팅노드들에게 통지(advertise)했는지 여부를 알리기 위한 플래그(IOP flag), 해당 포워딩정보를 로컬 포워딩테이블에 저장했는지 여부를 알리기 위한 플래그(FT flag)를 포함한다. 이러한 노드별 관리데이터들은 어그리게이션 트리(aggregation tree)가 업데이트될 경우 같이 업데이트하여 관리한다.

<42> 상기 포워딩정보의 타입정보(Type)는 IOP 내에서 해당 포워딩정보를 생성한 라우팅 프로토콜의 종류(예컨대, BGP, SDPF, RIP 등)를 말한다. 가상노드는 실제 포워딩정보와 구별하기 위한 별도의 타입(AGG)을 지정하여 사용하며, 다른 타입(예컨대, BGP, SDPF, RIP 등)의 포워딩정보에 대해서 가장 낮은 우선순위를 가진다. 이는 가상노드는 가상의

포워딩정보이므로 라우팅 프로토콜들이 제공하는 실제의 포워딩정보들에게 높은 우선순위를 부여하기 위함이다.

<43> 도 4b는 본 발명의 일 실시 예에 따라 포워딩정보를 관리하기 위해 생성된 어그리게이션 트리(aggregation tree) 구조에 대한 예시도이다. 도 4b의 예에서, 빗금이 있는 동그라미로 표시된 노드만이 임의의 라우팅노드(RN)에서 관리하는 포워딩정보를 나타낸다. 도 4b를 참조하면, 해당 라우팅노드(RN)에는 3개의 포워딩정보(P1, P2, P3)가 관리되며, 그들 각각의 포워딩주소는 P1=X.X.X.1, P2=X.X.X.10, P3=X.X.X.11이다. 그런데, P2와 P3은 포워딩주소가 'X.X.X.1'인 가상의 노드(V1)에 의해 어그리게이션(aggregation)될 수 있고, P1과 V1은 다시 포워딩주소가 'X.X.X.'인 가상의 노드(V2)에 의해 어그리게이션(aggregation)될 수 있다. 따라서, 상기 라우팅노드(RN)는 가상의 노드(V2)만을 가상영역으로 통지(advertise)하면 된다. 한편, 나머지 다른 라우팅노드(RN)들은 상기 가상의 노드(V2) 하나만을 로컬 포워딩테이블에 저장함으로써 상기 3개의 포워딩정보(P1, P2, P3)를 모두 저장하는 것과 같은 효과를 얻을 수가 있다. 한편, 도 4b를 참조하면, 상기 가상노드의 포워딩주소는 그 가상노드의 자노드인 포워딩정보의 포워딩주소에서 최하위 1비트값을 제외한 값으로 설정하였다.

<44> 본 발명의 예에서 어그리게이션 트리(aggregation tree)의 부모노드와 자식노드의 사이에서만 어그리게이션(aggregation)을 수행하는 이유는, 본 발명의 성능 검증에 사용될 real BGP(border gateway protocol) core 라우팅 테이블의 라우팅 엔트리 분포가 도 4c에 도시된 바와 같이 포워딩주소정보의 길이(prefix legnth)가 '24'인 지점에 집중적으로 분포한다는 분석결과로부터 부모노드와 자식노드사이의 어그리게이션(aggregation)

만으로도 충분한 포워딩정보의 어그리게이션(aggregation) 효과와 함께 동적으로 포워딩 정보의 어그리게이션(aggregation)시 오버헤드(averhead)를 줄일 수 있기 때문이다.

<45> 도 5는 새롭게 추가된 포워딩정보를 본 발명의 일 실시 예에 따라 관리하는 방법에 대한 처리 흐름도이다. 도 5를 참조하면, 라우팅노드(RN)들을 다수개 포함하는 분산구조라우터에서 임의의 한 라우팅노드(RN)에 새로운 포워딩정보가 추가되면(S110), 해당 라우팅노드(RN)는 그 포워딩정보를 로컬 어그리게이션 트리(aggregation tree)에 추가한다(S120). 이는 상기 추가된 포워딩정보에 의거하여 어그리게이션 트리(aggregation tree)에 새로운 노드를 추가하는 것을 말한다. 이 때, 어그리게이션 트리(aggregatio tree)에 새롭게 추가된 노드는 도 4a에 예시된 바와 같은 데이터구조에 의해 그 속성이 결정된다.

<46> 그리고, 상기 과정(S110)에서 추가된 포워딩정보의 생성영역을 확인하여(S130), 그 생성영역에 따른 처리를 수행한다. 예컨대, 상기 과정(S110)에서 추가된 포워딩정보가 로컬영역에서 생성된 경우는 로컬영역정보 추가과정을 수행하고(S140), 상기 과정(S110)에서 추가된 포워딩정보가 가상영역에서 생성된 경우는 가상영역정보 추가과정을 수행한다(S150). 여기서,'로컬영역'은 각 라우팅 노드에 물리적으로 연결된 라우팅정보 및 서브-넷을 의미하고, '가상영역'은 라우팅 노드들간의 연결에 의해 가상적으로 형성된 네트웍 영역을 의미한다. 그런데, 이러한'가상영역'에서는 각 라우팅노드(RN)들 간의 식별을 위해 사설 IP주소(private IP address)를 이용하는 것이 일반적이다. 따라서, 상기 과정(S130)에서는 상기 과정(S110)에서 추가된 포워딩정보의 생성영역을 확인하기 위해 그 포워딩정보의 주소가 사설 IP 주소인지의 여부를 판단하는 것이 바람직하다. 예컨대, 상기 과정(S110)에서 추가된 포워딩정보의 주소가 사설 IP 주소인 경우 그 포워딩정보는

가상영역에서 생성된 것으로 판단하고, 상기 과정(S110)에서 추가된 포워딩정보의 주소가 사설 IP 주소가 아닌 경우 그 포워딩정보는 로컬영역에서 생성된 것으로 판단한다.

<47> 한편, 상기 로컬영역정보 추가과정(S140) 또는 가상영역정보 추가과정(S150)의 수행결과를 어그리게이션 트리(aggregation tree)에 업데이트한다. 예를 들어, 상기 과정(S120)에서 로컬 어그리게이션 트리(aggregation tree)에 추가된 노드의 정보를 상기 로컬영역정보 추가과정(S140)에서 가상영역으로 통지(advertise)한 후 로컬 포워딩테이블에 저장한 경우, IOP Flag(도 4a 참조) 및 FT Flag(도 4a 참조)값을 모두 'yes'로 변경한다. 한편, 상기 과정(S120)에서 로컬 어그리게이션 트리(aggregation tree)에 추가된 노드의 정보를 상기 로컬영역정보 추가과정(S140)에서 가상영역으로 통지(advertise)하지 않고 로컬 포워딩테이블에만 저장한 경우, FT Flag(도 4a 참조)값만을 'yes'로 변경한다.

<48> 도 6 및 도 7에는 이러한 로컬영역정보 추가과정(S140) 및 가상영역정보 추가과정(S150)에 대한 처리 흐름이 도시되어 있다.

<49> 도 5 및 도 6을 참조하면, 상기 로컬영역정보 추가과정(S140)은 상기 과정(S120)에서 추가된 노드의 부모노드가 어그리게이션 트리(aggregation tree)에 존재하는 지 그렇지 않은지를 판단하고(S141), 상기 과정(S120)에서 추가된 노드와 그 부모노드의 생성영역이 동일한지 다른지의 여부를 판단한다(S142). 즉, 상기 과정(S120)에서 추가된 노드가 동일한 라우팅노드(RN)에 의해 생성되었는지, 그렇지 않은지의 여부를 판단한다. 그리고, 그 판단결과에 의거하여 상기 과정(S110)에서 추가된 포워딩정보를 다른 라우팅노드들(other RNs)에게 통지(advertise)할 지의 여부를 결정한다.

<50> 예컨대, 상기 과정(S141)의 판단결과 상기 과정(S120)에서 추가된 노드의 부모노드가 어그리게이션 트리(aggregation tree)에 존재하고 상기 과정(S142)의 판단결과 상기 과정(S120)에서 추가된 노드와 그 부모노드가 동일한 라우팅노드(RN)에 의해 생성된 경우, 상기 과정(S110)에서 임의의 라우팅노드(RN)에 추가된 포워딩정보를 다른 라우팅노드들(other RNs)에게 통지(advertise)하지 않고 해당 라우팅노드(RN)의 로컬포워딩테이블(local forwarding table)에만 저장한다(S143). 이 경우가 로컬영역정보 추가시 어그리게이션에 의한 효과를 기대할 수 있는 경우이다. 즉, 이 경우 상기 라우팅노드(RN)가 추가된 포워딩정보를 다른 라우팅노드들(other RNs)에게 통지(advertise)하지 않음으로써 이를 위한 제어패킷(control-packet)의 전송량을 줄이고 알고리즘의 복잡도를 줄일 수 있는 효과가 있는 것이다.

<51> 또한, 상기 과정(S141)의 판단결과 상기 과정(S120)에서 추가된 노드의 부모노드가 어그리게이션 트리(aggregation tree)에 존재하고 상기 과정(S142)의 판단결과 상기 과정(S120)에서 추가된 노드와 그 부모노드가 서로 다른 라우팅노드(RN)에 의해 생성된 경우, 상기 과정(S110)에서 임의의 라우팅노드(RN)에 추가된 포워딩정보를 다른 라우팅노드(RN)들에게 통지(advertise)한 후(S145), 그 포워딩정보를 해당 라우팅 노드의 로컬 포워딩 테이블에 저장한다(S146).

<52> 한편, 상기 과정(S141)의 판단결과 상기 과정(S120)에서 추가된 노드의 부모노드가 어그리게이션 트리(aggregation tree)에 존재하지 않는 경우, 상기 과정(S120)에서 추가된 노드의 부모노드를 생성한 후(S144), 상기 과정(S110)에서 임의의 라우팅노드(RN)에 추가된 포워딩정보를 다른 라우팅노드들(other RNs)에게 통지(advertise)하고(S145), 그 포워딩정보를 해당 상기 라우팅노드(RN)의 로컬 포워딩 테이블에 저장한다(S146).

<53> 이 때, 상기 부모노드는 가상의 포워딩정보를 나타내는 것으로서, 그 부모노드의 포워딩 주소정보는 상기 과정(S120)에서 추가된 노드의 포워딩 주소정보(Prefix)의 최하위 1비트값을 제외한 값으로 설정하고, 상기 부모노드를 생성한 라우팅노드(RN)는 상기 과정(S120)에서 추가된 노드를 생성한 라우팅노드(RN)으로 설정하는 것이 바람직하다.

<54> 도 5 및 도 7을 참조하면, 상기 가상영역정보 추가과정(S140)은 상기 과정(S120)에서 추가된 노드의 부모노드가 어그리게이션 트리(aggregation tree)에 존재하는 지 그렇지 않은지를 판단하고(S151), 상기 과정(S120)에서 추가된 노드와 그 부모노드의 생성영역이 동일한지 다른지의 여부를 판단한다(S152). 즉, 상기 과정(S120)에서 추가된 노드가 동일한 라우팅노드(RN)에 의해 생성되었는지, 그렇지 않은지의 여부를 판단한다. 그리고, 그 판단결과에 의거하여 상기 과정(S110)에서 추가된 포워딩정보를 해당 라우팅노드(RN)의 로컬 포워딩테이블(local forwarding table)에 저장할 지의 여부를 판단한다.

<55> 예컨대, 상기 과정(S151)의 판단결과 상기 과정(S120)에서 추가된 노드의 부모노드가 어그리게이션 트리(aggregation tree)에 존재하고 상기 과정(S152)의 판단결과 상기 과정(S120)에서 추가된 노드와 그 부모노드가 동일한 라우팅노드에 의해 생성된 경우, 상기 과정(S110)에서 새롭게 추가된 포워딩정보를 해당 라우팅노드(RN)의 로컬 포워딩테이블(local forwarding table)에 저장하지 않는다. 이 경우가 가상영역정보 추가시 어그리게이션에 의한 효과를 기대할 수 있는 경우이다. 즉, 다른 라우팅노드(other RN)에 새로운 포워딩정보가 추가되더라도, 그 포워딩정보를 대표하는 가상의 노드(부모노드)가 해당 라우팅노드(RN)에 이미 존재하는 경우 새로운 포워딩정보를 로컬포워딩테이블(local forwarding table)에 추가하지 않음으로써 해당 라우팅노드(RN)의 로컬포워딩테이블(local forwarding table)의 크기를 줄일 수 있는 효과가 있는 것이다.

<56> 또한, 상기 과정(S151)의 판단결과 상기 과정(S120)에서 추가된 노드의 부모노드가 어그리게이션 트리(aggregation tree)에 존재하고 상기 과정(S152)의 판단결과 상기 과정(S120)에서 추가된 노드와 그 부모노드가 서로 다른 라우팅노드에 의해 생성된 경우, 상기 과정(S110)에서 새롭게 추가된 포워딩정보를 해당 라우팅노드(RN)의 로컬 포워딩 테이블(local forwarding table)에 저장한다(S153).

<57> 한편, 상기 과정(S151)의 판단결과 상기 과정(S120)에서 추가된 노드의 부모노드가 어그리게이션 트리(aggregation tree)에 없는 경우는 상기 과정(S120)에서 추가된 노드의 부모노드를 생성한 후(S154), 상기 과정(S154)에서 생성된 부모노드를 로컬 포워딩 테이블(local forwarding table)에 저장한다(S155). 이 때 부모노드를 생성하는 과정(S154)은 도 6의 과정(S144) 설명시 언급되었으므로 생략한다.

<58> 도 8은 삭제된 포워딩정보를 본 발명의 일 실시 예에 따라 관리하는 방법에 대한 처리 흐름도이다. 도 8을 참조하면, 라우팅 노드들을 다수개 포함하는 분산구조라우터에서 임의의 한 라우팅노드(RN)에서 포워딩정보가 삭제되면(S210), 그 라우팅노드(RN) 기설정된 어그리게이션 트리(aggregation tree)로부터 상기 과정(S210)에서 삭제된 포워딩정보에 해당되는 노드정보를 추출하여 상기 포워딩정보가 상기 라우팅노드의 로컬 영역에서 생성되었는지의 여부를 판단한다(S220). 이 때, 상기 과정(S210)에서 삭제된 포워딩정보의 생성영역을 판단하기 위해, 도 5의 과정(S130)에서도 예시된 바와 같이, 상기 과정(S210)에서 포워딩정보의 주소정보가 사설 IP 주소인지의 여부를 판단하는 것이 바람직하다. 예컨대, 상기 과정(S210)에서 삭제된 포워딩정보의 주소가 사설 IP 주소인 경우 그 포워딩정보는 가상영역에서 생성된 것으로 판단하고, 상기 과정(S110)에서 추가된

포워딩정보의 주소가 사설 IP 주소가 아닌 경우 그 포워딩정보는 로컬영역에서 생성된 것으로 판단한다.

<59> 그리고, 그 판단결과(S220)에 의거하여 상기 과정(S210)에서 삭제된 포워딩정보의 생성영역에 따른 처리를 수행한다. 예컨대, 상기 과정(S210)에서 삭제된 포워딩정보가 로컬영역에서 생성된 경우는 로컬영역정보 삭제과정을 수행하고(S230), 상기 과정(S210)에서 삭제된 포워딩정보가 가상영역에서 생성된 경우는 가상영역정보 삭제과정을 수행한다(S240).

<60> 도 9 및 도 10에는 이러한 로컬영역정보 삭제과정(S230) 및 가상영역정보 삭제과정(S240)에 대한 처리 흐름이 도시되어 있다.

<61> 도 8 및 도 9를 참조하면, 상기 로컬영역정보 삭제과정(S230)은 상기 과정(S210)에서 삭제된 포워딩정보가 가상영역으로 통지(advertise)되었는지의 여부를 먼저 확인하고(S231) 그 결과에 의거하여 해당 포워딩정보의 삭제정보를 가상영역으로 통지할지의 여부를 결정한다. 이 때, 상기 확인결과(S231) 상기 과정(S210)에서 삭제된 포워딩정보가 상기 라우팅노드(RN)의 로컬 영역에서 생성되고 그 포워딩정보가 다른 라우팅노드들(other RNs)에게 통지된 경우에만 상기 과정(S210)에서 삭제된 포워딩정보의 삭제정보를 다른 라우팅노드들(other RNs)에게 통지한다(S232).

<62> 그리고, 그 삭제된 정보에 해당되는 노드(삭제대상노드)의 형제노드를 어그리게이션 트리(aggregation tree)에서 검색한다(S233). 이는 디스어그리게이션(disaggregation)을 수행하기 위함이다. 상기 검색(S233) 결과 삭제대상노드의 형제노드가 어그리게이션 트리(aggregation tree)에 존재하면(S234), 그 형제노드정보를 다른 라우팅노드에게 통지(advertise)한다(S235). 즉, 그 형제노드정보를 가상영역으로 통지한

다. 이 때, 그 형제노드정보는 삭제대상노드에 대한 삭제정보가 가상영역으로 통지된 후 소정 시간이 흐른 뒤에 통지되므로 이를 지연리포트(delayed report)라 명명한다. 이처럼 가상영역으로 상기 과정(S210)에서 삭제된 포워딩정보에 대한 삭제정보를 모두 전송한 후에 해당 어그리게이션 트리 (aggregation tree)및 해당 라우팅노드의 로컬 포워딩 테이블(local forwarding tagle)에서 삭제대상노드를 삭제한다(S236).

<63> 한편, 상기 검색(S233)결과, 삭제대상노드의 형제노드가 없으면 어그리게이션 트리(aggregation tree) 및 로컬 포워딩테이블(local forwarding table)에서 삭제대상노드 및 그 부모노드를 삭제한다(S237).

<64> 도 10은 본 발명의 일 실시 예에 따라 가상영역정보를 삭제하는 과정에 대한 처리 흐름도이다.

<65> 도 8 및 도 10을 참조하면, 상기 가상역정보 삭제과정(S240)은 우선, 상기 삭제된 포워딩정보에 해당되는 노드(삭제대상노드)를 어그리게이션 트리(aggregation tree)에서 삭제한다(S241). 그리고, 가상영역으로부터 그 삭제된 노드의 형제노드정보(예컨대, delayed report)를 수신한 경우(S242), 그 형제 노드를 상기 어그리게이션 트리 (aggregation tree)에 추가한다(S243). 즉, 상기 형제노드정보(delayed report)에 의해 디스어그리게이션(disaggregation)을 수행한다. 또한, 상기 과정(S243)에서 추가된 노드의 부모노드가 어그리게이션 트리(aggregation tree)에 있는지 없는지를 확인하고 (S244), 상기 과정(S243)에서 추가된 노드와 그 부모노드를 생성한 라우팅노드(RN)가 동일한지 그렇지 않은지를 확인한 후(S245) 그 결과에 의해 상기 과정(S243)에서 추가된 노드 정보를 로컬 포워딩 테이블(local forwarding table)에 저장할 것 인지의 여부를 결정한다.

<66> 예컨대, 상기 과정(S244)의 판단결과 상기 과정(S243)에서 추가된 노드의 부모노드가 어그리게이션 트리(aggregation tree)에 존재하고 상기 과정(S245)의 판단결과 상기 과정(S243)에서 추가된 노드와 그 부모노드가 동일한 라우팅노드에 의해 생성된 경우, 상기 과정(S243)에서 추가된 노드정보를 해당 라우팅노드(RN)의 로컬 포워딩테이블(local forwarding table)에 저장하지 않는다.

<67> 또한, 상기 과정(S244)의 판단결과 상기 과정(S243)에서 추가된 노드의 부모노드가 어그리게이션 트리(aggregation tree)에 존재하고 상기 과정(S245)의 판단결과 상기 과정(S243)에서 추가된 노드와 그 부모노드가 서로 다른 라우팅노드에 의해 생성된 경우, 상기 과정(S243)에서 추가된 노드정보를 해당 라우팅노드(RN)의 로컬 포워딩 테이블(local forwarding table)에 저장한다(S246).

<68> 한편, 상기 과정(S244)의 판단결과 상기 과정(S243)에서 추가된 노드의 부모노드가 어그리게이션 트리(aggregation tree)에 없는 경우는 상기 과정(S243)에서 추가된 노드의 부모노드를 생성한 후(S247), 상기 과정(S247)에서 생성된 부모노드를 로컬 포워딩 테이블(local forwarding table)에 저장한다(S248). 이 때 부모노드를 생성하는 과정(S247)은 도 6의 과정(S144) 설명시 언급되었으므로 생략한다.

<69> 도 11a 및 도 11b는 라우팅 테이블에 새롭게 추가된 라우팅 정보를 관리하는 방법을 도식화하여 설명한 도면이다. 도 11a는 IOP#1(210)에 포워딩주소(Prefix)가 '3'인 새로운 포워딩정보가 추가된 경우 이를 IOP#1의 어그리게이션 트리(Aggregation tree) 및 포워딩 테이블(Forwarding table)에 추가하고, 이를 IOP#n에게 통지하는 과정을 도식화하여 설명한 도면이고, 도 11b는 IOP#1(210)에 포우딩주소(Prefix)가 '4'인 새로운 포워

딩정보가 추가된 경우 이를 IOP#1의 어그리게이션 트리 및 포워딩 테이블에 추가하는 과정을 도식화하여 설명한 도면이다.

<70> 도 11a 및 도 11b의 어그리게이션 트리의 IOP 영역의 빗금은 해당 포워딩정보를 가상영역으로 통지하였음을 나타내고, FE는 해당 포워딩정보를 로컬 포워딩테이블에 저장하였음을 나타내는 표시이다.

<71> 우선, 도 11a를 참조하면, IOP#1(210)에 아무 정보도 등록되지 않은 상태에서 포워딩주소(Prefix)가 '3'인 새로운 포워딩정보가 추가된 경우 IOP#1(210)의 제어부는 그 정보를 IOP#1(210)의 어그리게이션 트리 및 포워딩 테이블에 추가하고, 포워딩주소(Prefix)가 '1'인 가상의 부모노드를 생성한다. 이 때, 포워딩주소(Prefix)가 '3'인 노드와 포워딩주소(Prefix)가 '1'인 노드들은 모두 IOP#1(210)에서 생성되었으므로, 어그리게이션 트리의 Source IOP 영역에는 모두 '1'이 기재되었다. 또한, IOP#1(210)은 그 삽입정보를 IOP#n(290)에게 통지한다. 이 때, IOP#1(210)의 포워딩 테이블에는 실제 포워딩정보인 포워딩주소(Prefix)가 '3'인 노드정보가 저장되었다.

<72> 한편, IOP#1(210)로부터 삽입정보를 통지 받은 IOP#n(290)은 포워딩주소(Prefix)가 '3'인 노드와 포워딩주소(Prefix)가 '1'인 노드들을 생성하여 IOP#n(290)의 어그리게이션 트리에 저장한다. 이 때, 포워딩주소(Prefix)가 '3'인 노드와 포워딩주소(Prefix)가 '1'인 노드들은 모두 IOP#1(210)에서 최초로 생성되었으므로, IOP#n(290)의 Source IOP 영역에도 '1'이 기재되었다. 그런데, 여기서 IOP#n(290)의 포워딩테이블에는 포워딩주소(Prefix)가 '3'인 노드의 부모노드인 포워딩주소(Prefix)가 '1'인 노드정보가 저장되었다. 이는 IOP#n(290)가 포워딩주소(Prefix) '3'인 노드의 포워딩정보를 가상영역으로부터 제공받았음을 나타낸다.

<73> 도 11b를 참조하면, IOP#1(210)에 포워딩주소(Prefix)가 '3'인 포워딩정보가 추가된 다음, 포워딩주소(Prefix)가 '4'인 포워딩정보가 추가된 경우, 어그리게이션이 수행되어 포워딩주소(Prefix)가 '4'인 포워딩정보는 IOP#n(290)으로 통지(advertise)하지 않아도 됨을 알 수 있다. 따라서, IOP#n(290)의 로컬영역에 물린 인터페이스 장비들은 IOP#n(290)의 포워딩테이블의 정보에 의거하여 IOP#1(210)을 찾아가고, IOP#1(210)의 포워딩테이블의 정보에 의거하여 포워딩주소(Prefix)가 '3'또는 포워딩주소(Prefix)가 '4'인 인터페이스 장비로 라우팅이 가능한 것이다.

<74> 도 12a 및 도 12b는 라우팅 테이블에서 삭제된 라우팅 정보를 관리하는 방법을 도식화하여 설명한 도면이다.

<75> 도 12a는 도 11b와 같이 포워딩주소(Prefix)가 '3'인 포워딩정보 및 포워딩주소(Prefix)가 '4'인 포워딩정보가 IOP#1(210)의 로컬영역에 연결되어 있다가, 포워딩주소(Prefix)가 '4'인 포워딩정보가 삭제된 경우에 대한 예를 나타낸다. 도 11b의 예에서 포워딩주소(Prefix)가 '4'인 포워딩정보는 가상영역으로 통지되지 않은 상태이므로, 도 12a의 예에서는 그 삭제정보를 IOP#1(210)의 어그리게이션 트리 및 포워딩 테이블에서만 적용하면 된다. 즉, 가상영역으로 통지할 필요가 없다.

<76> 한편, 도 12b는 도 11b와 같이 포워딩주소(Prefix)가 '3'인 포워딩정보 및 포워딩주소(Prefix)가 '4'인 포워딩정보가 IOP#1의 로컬영역에 연결되어 있다가, 포워딩주소(Prefix)가 '3'인 포워딩정보가 삭제된 경우에 대한 예이다. 도 11b의 예에서 포워딩주소(Prefix)가 '3'인 포워딩정보는 가상영역으로 통지된 상태이므로, 도 12b의 예에서는 그 삭제정보를 가상영역을 통해 IOP#n(290) 측으로 통지하여야 한다. 그리고, IOP#1(210)의 어그리게이션 트리에서 포워딩주소(Prefix)가 '3'인 노드의 형제노드를 검

색하여 그 형제노드(포워딩주소(Prefix)가 '4'인 노드)의 정보를 지연리포트(delayed report)로 제공한다.

<77> 그러면, 이 지연리포트를 제공받은 IOP#n(290)은 포워딩주소(Prefix)가 '3'인 포워딩정보를 삭제하고, 포워딩주소(Prefix)가 '4'인 포워딩정보를 새로이 추가한다.

<78> 도 13a 내지 도 13d는 본 발명의 일 실시 예에 따른 포워딩정보 동적 관리방법의 효과를 입증하기 위한 테스트 결과들이다.

<79> 본 발명에서는 이러한 테스트 결과를 얻기 위해 1개의 SWM과 2개의 IOP를 갖춘 Galaxy 시스템과 에서 제공되는 core BGP 라우터의 라우팅 테이블 엔트리를 이용하여 53,000개의 라우팅 엔트리를 삽입하는 경우와 이중의 일정 비율을 플랩(flap)하는 테스트를 수행하였다.

<80> 도 13a는 어그리게이션(aggregation) 효과가 삽입되는 엔트리의 순서에 영향을 받는지를 측정하기 위한 테스트 결과를 도시한 도면으로서, 10번에 걸쳐서 오름차순으로 정렬된 샘플 엔트리의 순서를 무작위로 변경하면서 삽입해본 결과를 나타낸다. 도 13a를 참조하면, 상기 테스트 결과 제어 패킷의 최대치와 최소치의 차이는 10개 이내였으며, 포워딩 엔트리 개수는 60개 이내를 나타내고 있다. 따라서, 본 발명의 어그리게이션 성능은 엔트리들의 삽입순서에는 거의 영향을 받지 않는 다는 것을 알 수 있다.

<81> 도 13b는 0~70%까지 53,000개의 BGP 라우팅 엔트리를 플랩(flap)시키면서 이 때 IOP간의 라우팅테이블 동기화를 위해 전송되는 제어패킷 전송량의 측정 결과를 도시한 도면이다. 도 13b를 참조하면, 본 발명을 적용한 결과 약 24~32%의 제어패킷의 감소 효과가 있었음을 알 수 있다.

<82> 도 13c는 53,000개의 core BGP 라우팅 엔트리를 각각 0~70% 삭제하면서 포워딩 테이블에 존재하는 포워딩 엔트리의 개수를 측정한 결과를 도시한 도면이다. 도 13c를 참조하면, 이 경우 core BGP 엔트리를 단순 삽입한 경우 0%, 삭제인 경우 약 20%의 포워딩 테이블 엔트리 감소 효과가 있었음을 알 수 있다.

<83> 도 13d는 플랩 엔트리의 개수에 따른 컨버젼스 시간(convergence time) 결과를 나타낸 도면이다. 도 13d를 참조하면, 본 발명의 경우 컨버젼스 시간(convergence time)이 줄어들었음을 알 수 있다.

<84> 도 14a 내지 도 14b는 본 발명의 일 실시 예에 따른 포워딩정보를 동적으로 관리하는 방법에 대한 알고리즘 예를 나타낸 도면이다. 도 14a에는 본 발명의 일 실시 예에 따른 포워딩정보를 동적으로 삽입하는 과정에 대한 알고리즘을 나타내고, 도 14b는 본 발명의 일 실시 예에 따른 포워딩정보를 동적으로 삭제하는 과정에 대한 알고리즘을 나타내고, 도 14c 내지 도 14j는 도 14a 및 도 14b의 알고리즘을 구현하기 위한 서브-알고리즘들을 나타낸다.

<85> 상술한 본 발명의 설명에서는 구체적인 실시 예에 관해 설명하였으나, 여러 가지 변형이 본 발명의 범위에서 벗어나지 않고 실시할 수 있다. 따라서 본 발명의 범위는 설명된 실시 예에 의하여 정할 것이 아니고 특허청구범위와 특허청구범위의 균등한 것에 의해 정해 져야 한다.

【발명의 효과】

<86> 상술한 바와 같이 본 발명은 분산구조라우터에서 포워딩정보의 추가 또는 삭제에 응답하여 그 포워딩정보를 어그리게이션(aggregation) 또는 디스어그리게이션(disaggregation)함으로써, 분산구조라우터의 각 라우팅노드들에서 관리하는 포워딩테이블의 크기를 줄일 수 있다. 또한, 분산구조라우터에서 포워딩테이블을 업데이트하기 위해 전송되는 제어-패킷(control packet)의 전송량을 줄여 내부 트래픽을 줄일 수 있다는 효과가 있다.

【특허청구범위】

【청구항 1】

라우팅 노드들을 다수개 포함하는 분산구조라우터에서 포워딩정보를 관리하는 방법에 있어서,

임의의 한 라우팅 노드에 새로운 포워딩정보가 삽입되면, 그 포워딩정보에 의거하여 어그리게이션 트리를 구성 및 변경하는 제1 과정과,

상기 제1 과정에서 삽입된 포워딩정보가 상기 라우팅 노드의 로컬 영역에서 생성되었는지의 여부를 판단하는 제2 과정과,

상기 제2 과정의 판단결과 상기 포워딩정보가 상기 라우팅 노드의 로컬 영역에서 생성된 경우, 상기 어그리게이션 트리를 분석하여 상기 포워딩정보를 다른 라우팅 노드들에게 통지(advertise)할 지의 여부를 판단하는 제3 과정과,

상기 제3 과정의 판단결과에 의거하여 상기 포워딩정보를 다른 라우팅 노드들에게 통지(advertise)한 후 그 포워딩정보를 해당 라우팅 노드의 로컬 포워딩테이블에 저장하는 제4 과정과,

상기 제2 과정의 판단결과 상기 포워딩정보가 상기 라우팅 노드의 로컬 영역에서 생성되지 않은 경우, 상기 어그리게이션 트리를 분석하여 상기 포워딩정보를 해당 라우팅 노드의 로컬 포워딩테이블에 저장할 지의 여부를 판단하는 제5 과정과,

상기 제5 과정의 판단결과에 의거하여 상기 포워딩정보를 해당 라우팅 노드의 로컬 포워딩테이블에 저장하는 제6 과정을 포함하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 2】

제1항에 있어서, 상기 제1 과정은

상기 포워딩정보가 포워딩될 주소정보(Prefix) 및 그 포워딩정보가 생성된 라우팅 노드 정보에 의거하여 그 포워딩정보를 어그리게이션 트리의 새로운 노드로 추가하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 3】

제2항에 있어서, 상기 어그리게이션 트리의 각 노드들은

그 노드들의 속성으로

해당 포워딩정보가 포워딩될 주소정보(Prefix), 그 포워딩될 주소의 길이정보(Length), 포워딩정보의 타입정보(Type), 포워딩정보를 생성한 라우팅노드 정보(Source IOP), 해당 포워딩정보를 다른 라우팅노드들에게 통지(advertise)했는지 여부를 알리기 위한 플래그(IOP flag), 해당 포워딩정보를 로컬 포워딩테이블에 저장했는지 여부를 알리기 위한 플래그(FT flag)를 포함하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 4】

제3항에 있어서,

상기 제4 과정 및 상기 제6 과정 중 어느 한 과정 수행 후 그 수행결과를 상기 어그리게이션 트리의 노드들의 속성으로 업데이트하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 5】

제2항에 있어서, 상기 제2 과정은

상기 제1 과정에서 어그리게이션 트리에 추가된 노드 정보에 의거하여, 그 포워딩 정보가 상기 라우팅 노드의 로컬 영역에서 생성되었는지의 여부를 판단하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 6】

제2항에 있어서, 상기 제3 과정은

상기 제1 과정에서 추가된 노드의 부모노드가 상기 어그리게이션 트리에 존재하는지의 여부를 판단하는 제3-1 과정과,

상기 추가된 노드와 그 부모노드가 동일한 라우팅 노드에 의해 생성되었는지의 여부를 판단하는 제3-2 과정과,

상기 제3-1 과정 및 제3-2 과정의 판단결과에 의거하여 상기 포워딩정보를 다른 라우팅 노드들에게 통지(advertise)할 지의 여부를 판단하는 제3-3 과정을 포함하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 7】

제6항에 있어서, 상기 제3-3 과정은

상기 제3-1 과정의 판단결과 상기 추가된 노드의 부모노드가 상기 어그리게이션 트리에 존재하고 상기 제3-2 과정의 판단결과 상기 추가된 노드와 그 부모노드가 동일한 라우팅 노드에 의해 생성된 경우, 상기 포워딩정보를 다른 라우팅 노드들에게 통지(advertise)하지 않고 해당 라우팅 노드의 로컬 포워딩 테이블에만 저장하는 것으로 판단하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 8】

제6항에 있어서, 상기 제3-3 과정은

상기 제3-1과정의 판단결과 상기 추가된 노드의 부모노드가 상기 어그리게이션 트리에 존재하고 상기 제3-2 과정의 판단결과 상기 추가된 노드와 그 부모노드가 서로 다른 라우팅 노드에 의해 생성된 경우, 상기 포워딩정보를 다른 라우팅 노드들에게 통지(advertise)한 후 그 포워딩정보를 해당 라우팅 노드의 로컬 포워딩 테이블에 저장하는

것으로 판단하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 9】

제6항에 있어서, 상기 제3-3 과정은

상기 제3-1 과정의 판단결과 상기 추가된 노드의 부모노드가 상기 어그리게이션 트리에 존재하지 않는 경우, 상기 추가된 노드의 부모노드를 생성한 후 상기 포워딩정보를 다른 라우팅 노드들에게 통지(advertise)하고 그 포워딩정보를 해당 라우팅 노드의 로컬 포워딩 테이블에 저장하는 것으로 판단하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 10】

제9항에 있어서, 상기 제3-3 과정은

상기 추가된 노드의 포워딩 주소정보(Prefix)의 최하위 1비트값을 제외한 값을 포워딩 주소정보로 가지는 노드를 상기 새로운 노드의 부모노드로 생성하고, 상기 추가된 노드가 생성된 라우팅 노드 정보에 의거하여 그 부모노드가 생성된 라우팅 노드정보를 설정하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법

.

【청구항 11】

제2항에 있어서, 상기 제5 과정은

상기 제1 과정에서 추가된 노드의 부모노드가 상기 어그리게이션 트리에 존재하는지의 여부를 판단하는 제5-1 과정과,

상기 추가된 노드와 그 부모노드가 동일한 라우팅 노드에 의해 생성되었는지의 여부를 판단하는 제5-2 과정과,

상기 제5-1 과정 및 제5-2 과정의 판단결과에 의거하여 상기 포워딩정보를 해당 라우팅 노드의 로컬 포워딩테이블에 저장할 지의 여부를 판단하는 제5-3 과정을 포함하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 12】

제11항에 있어서, 상기 제5-3 과정은

상기 제5-1 과정의 판단결과 상기 추가된 노드의 부모노드가 상기 어그리게이션 트리에 존재하고 상기 제5-2 과정의 판단결과 상기 추가된 노드와 그 부모노드가 동일한 라우팅 노드에 의해 생성된 경우, 상기 포워딩정보를 해당 라우팅 노드의 로컬 포워딩테이블에 저장하지 않는 것으로 판단하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 13】

제11항에 있어서, 상기 제5-3 과정은

상기 제5-1과정의 판단결과 상기 추가된 노드의 부모노드가 상기 어그리게이션 트리에 존재하고 상기 제5-2 과정의 판단결과 상기 추가된 노드와 그 부모노드가 서로 다른 라우팅 노드에 의해 생성된 경우, 상기 포워딩정보를 해당 라우팅 노드의 로컬 포워딩 테이블에 저장하는 것으로 판단하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 14】

라우팅 노드들을 다수개 포함하는 분산구조라우터에서 포워딩정보를 관리하는 방법에 있어서,

임의의 한 라우팅 노드에서 포워딩정보가 삭제되면, 그 라우팅 노드 내에 기 설정된 어그리게이션 트리로부터 그 포워딩정보에 해당되는 노드정보를 추출하여 상기 포워딩정보가 상기 라우팅 노드의 로컬 영역에서 생성되었는지의 여부를 판단하는 제1 과정과,

상기 제1 과정의 판단결과 상기 포워딩정보가 상기 라우팅 노드의 로컬 영역에서 생성되고 그 포워딩정보가 다른 라우팅 노드들에게 통지된 경우에만 상기 포워딩정보의 삭제정보를 다른 라우팅 노드들에게 통지한 후 그 삭제된 포워딩정보를 상기 어그리게이션 트리 및 해당 라우팅 노드의 로컬 포워딩테이블에서 삭제하는 제2 과정과,

상기 제1 과정의 판단결과 상기 포워딩정보가 상기 라우팅 노드의 로컬 영역에서 생성되지 않은 경우 그 포워딩정보에 해당되는 노드를 상기 어그리게이션 트리에서 삭제

하는 제3 과정을 포함하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 15】

제14항에 있어서, 상기 제2 과정은

상기 포워딩정보의 삭제정보를 다른 라우팅 노드들에게 통지한 후 그 삭제된 포워딩정보에 해당되는 노드의 형제노드를 상기 어그리게이션 트리로부터 검출하는 제2-1 과정과,

상기 제2-1 과정의 검출결과에 의거하여 상기 형제노드 정보를 다른 라우팅 노드들에게 통지하는 제2-2 과정을 더 포함하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 16】

제15항에 있어서, 상기 제2 과정은

상기 제2-1 과정에서 상기 형제노드가 검출되지 않는 경우, 상기 삭제된 포워딩정보에 해당되는 노드의 부모노드를 삭제하는 제2-3 과정을 더 포함하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【청구항 17】

제14항에 있어서, 상기 제3 과정은

상기 삭제된 포워딩정보를 생성한 라우팅 노드로부터 그 포워딩정보에 해당되는 노드의 형제노드정보를 수신한 경우, 그 형제 노드를 상기 어그리게이션 트리에 추가하는 제3-1 과정과,

상기 어그리게이션 트리를 분석하고, 그 분석결과에 의거하여 상기 형제노드정보를 해당 라우팅 노드의 로컬 포워딩테이블에 저장하는 제3-2 과정을 포함하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

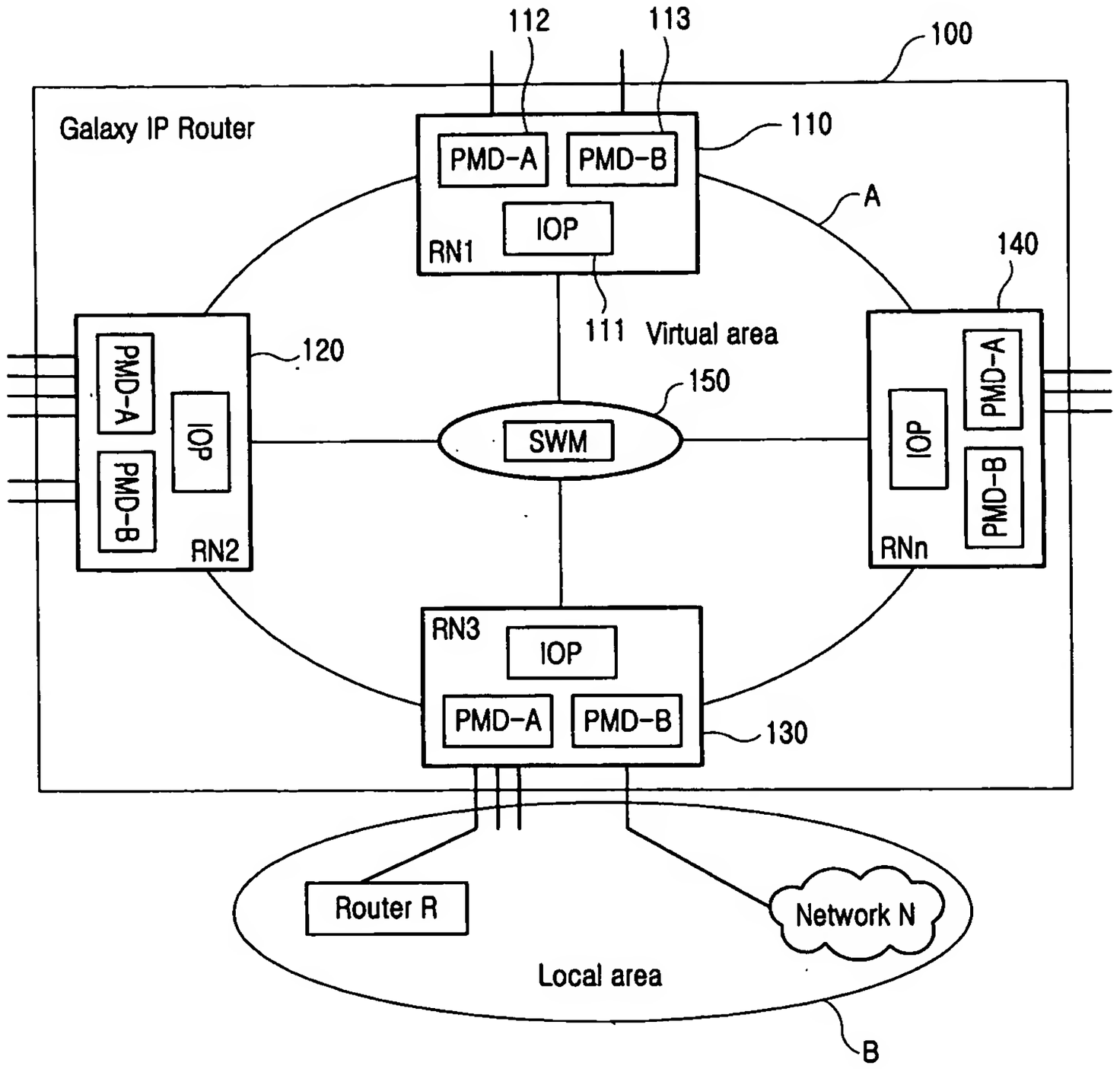
【청구항 18】

제17항에 있어서, 상기 제3-2 과정은

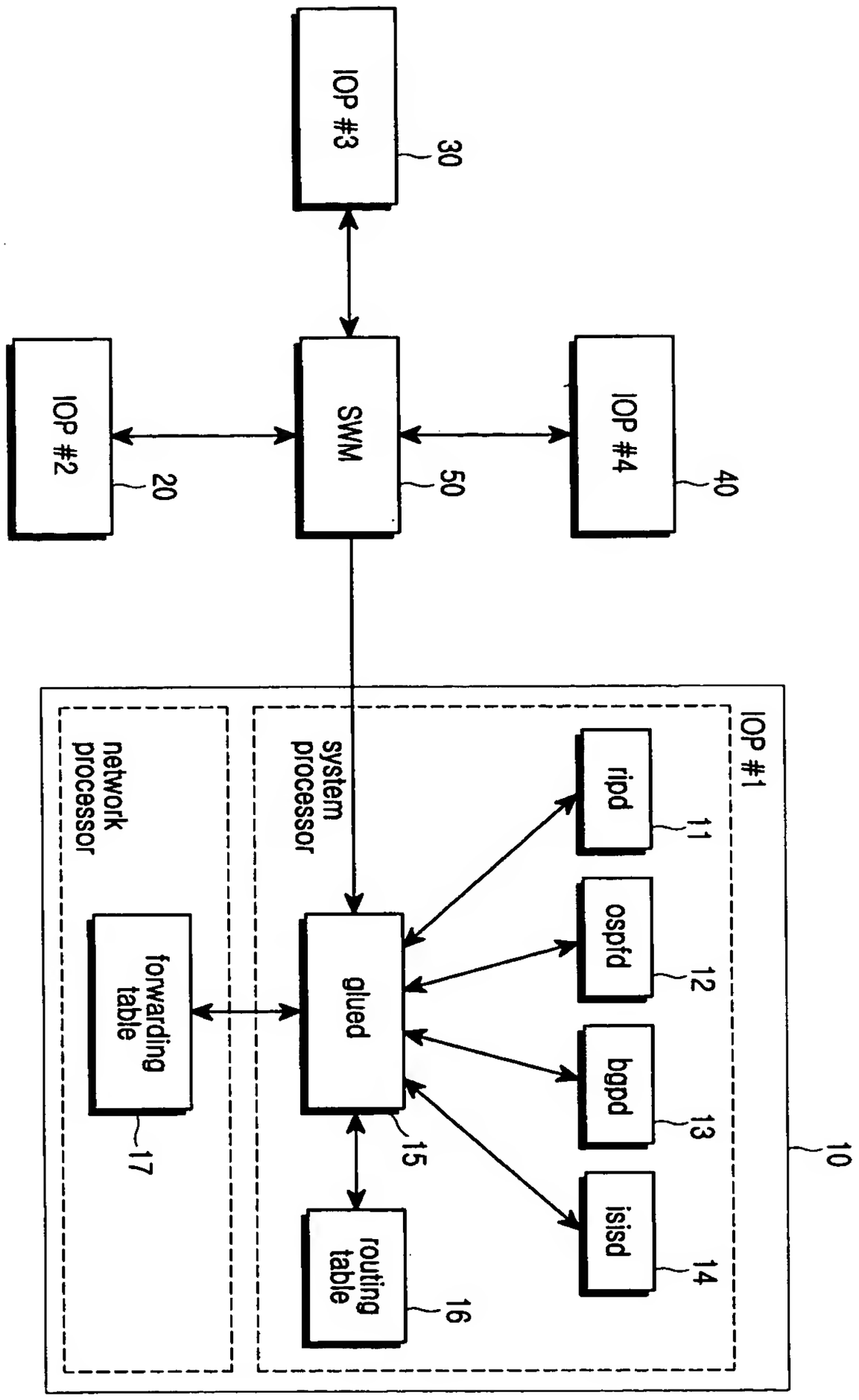
상기 제3-1 과정에서 추가된 노드의 부모노드가 상기 어그리게이션 트리에 존재하고, 상기 추가된 노드와 그 부모노드가 서로 다른 라우팅 노드에 의해 생성된 경우에 상기 추가된 노드정보를 로컬 포워딩테이블에 저장하는 것을 특징으로 하는 분산구조라우터에서 포워딩정보를 동적으로 관리하는 방법.

【도면】

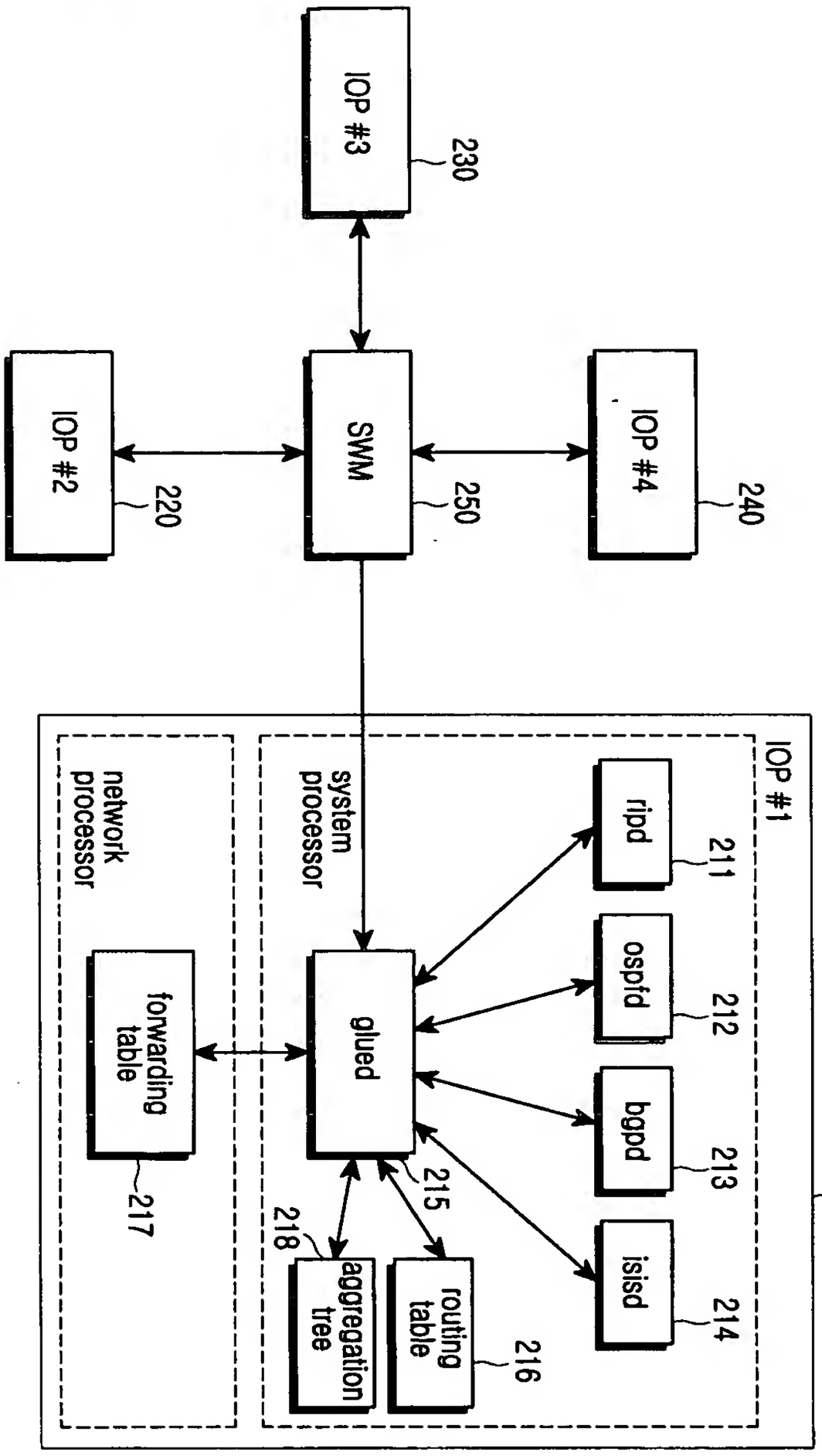
【도 1】


112
113
100
Galaxy IP Router
PMD-A
PMD-B
110
IOP
RN1
A
140
111 Virtual area
120
150
SWM
PMD-A
IOP
PMD-B
RN2
IOP
PMD-A
PMD-B
RNn
RN3
IOP
PMD-A
PMD-B
130
Router R
Network N
Local area
B

【도 2】


IOP #3
30
IOP #2
20
SWM
50
IOP #4
40
IOP #1
system processor
network processor
ripd
11
ospfd
12
bgpd
13
isisd
14
glued
15
routing table
16
forwarding table
17
10

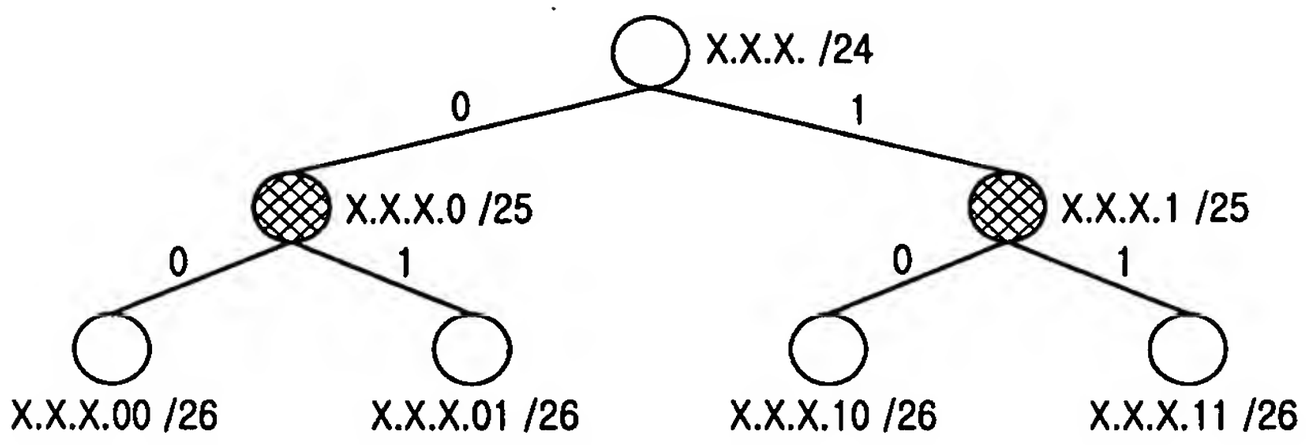
【도 3】



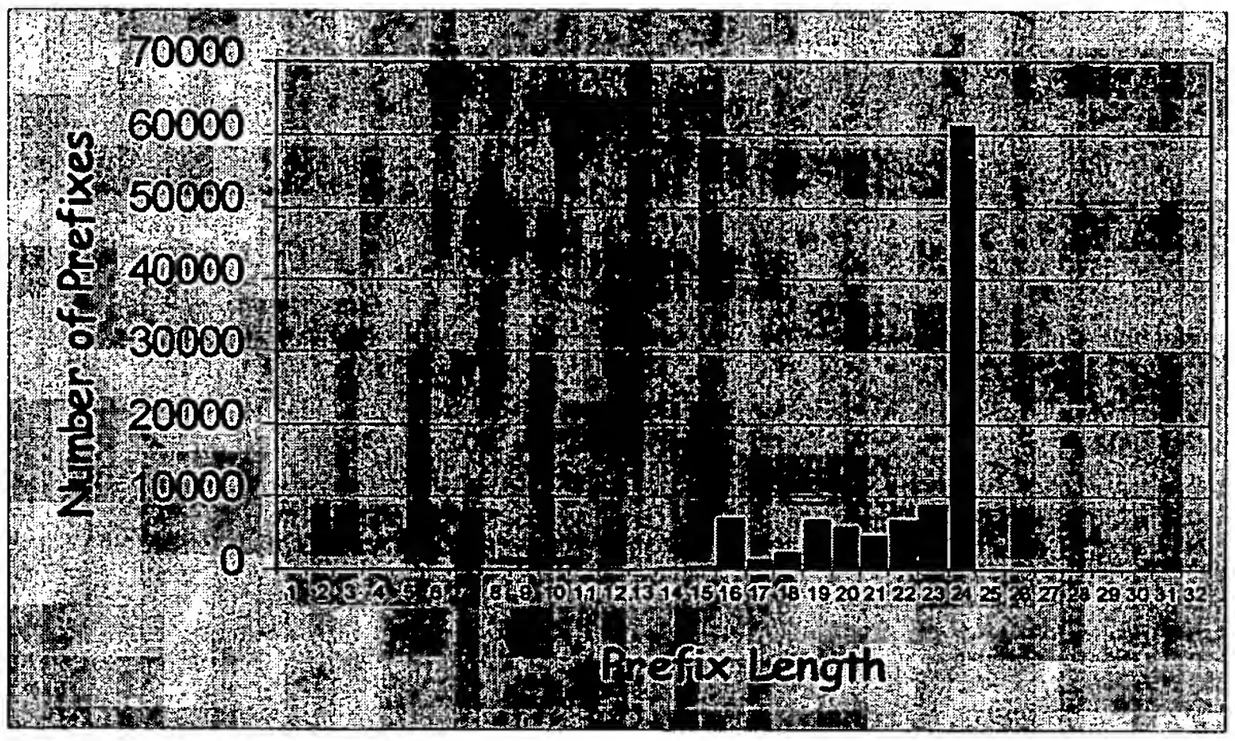
【도 4a】

Prefix	Length	Type	Source IOP	IOP Flag	FT Flag

【도 4b】

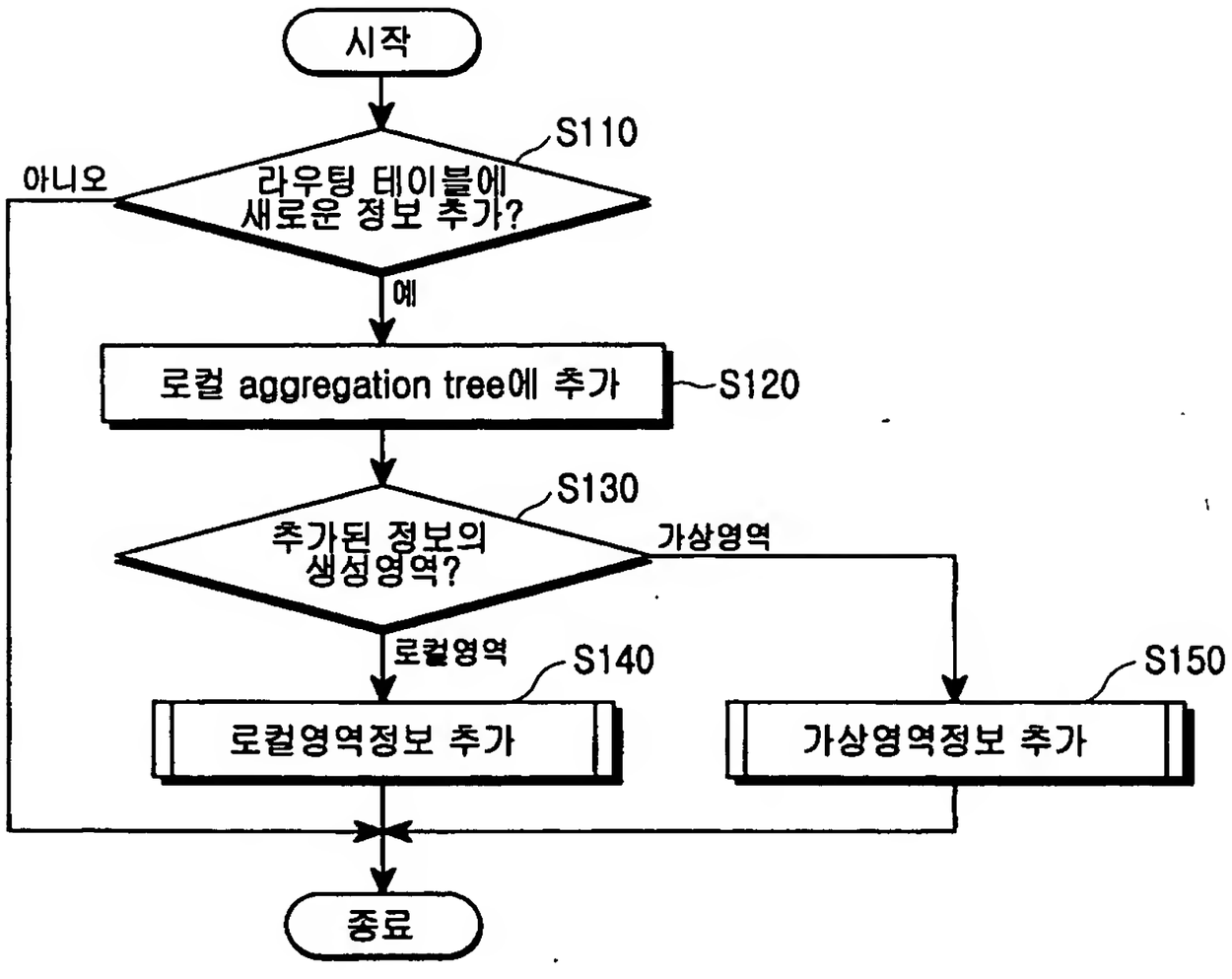


【도 4c】

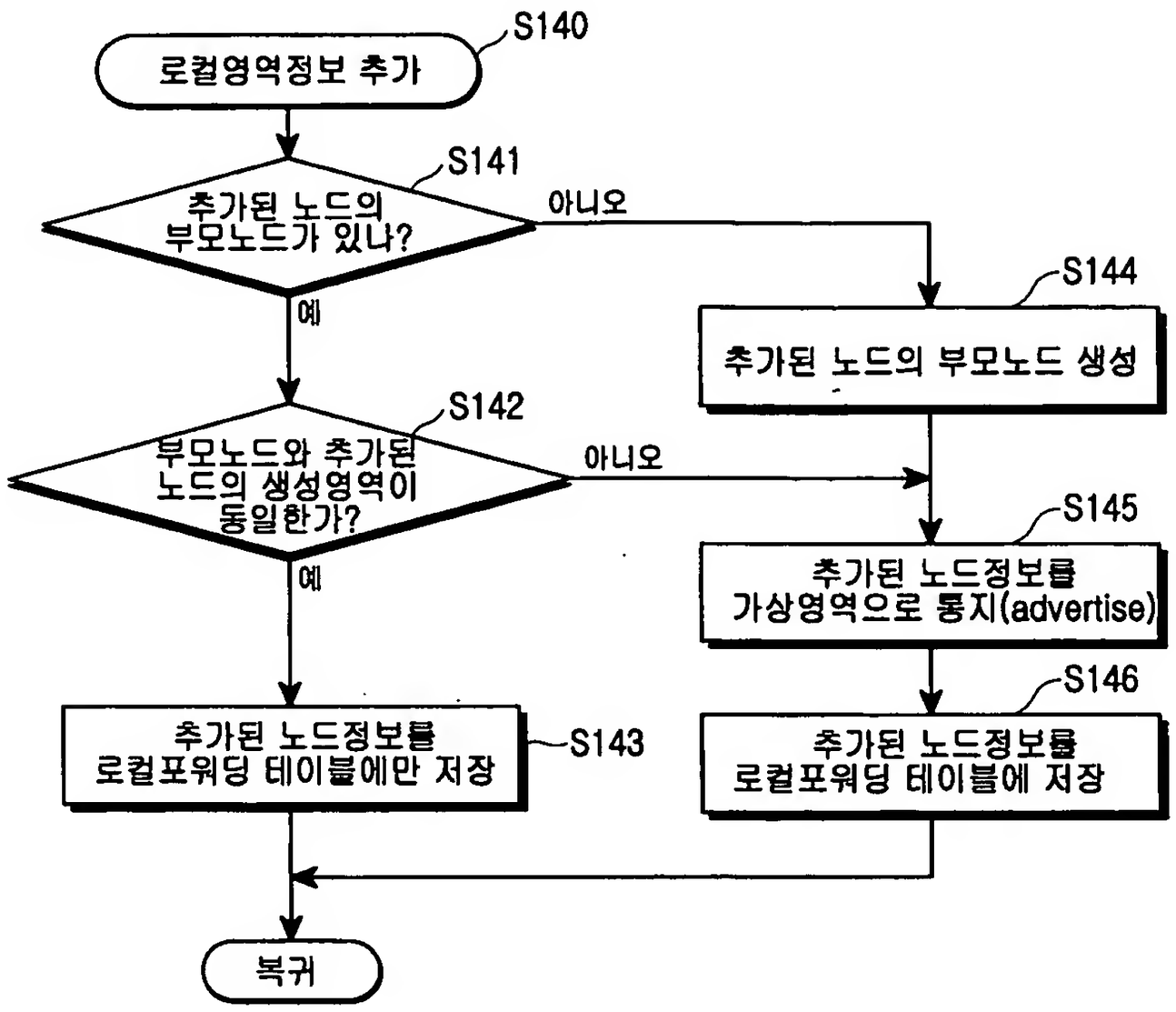


Prefix length distribution

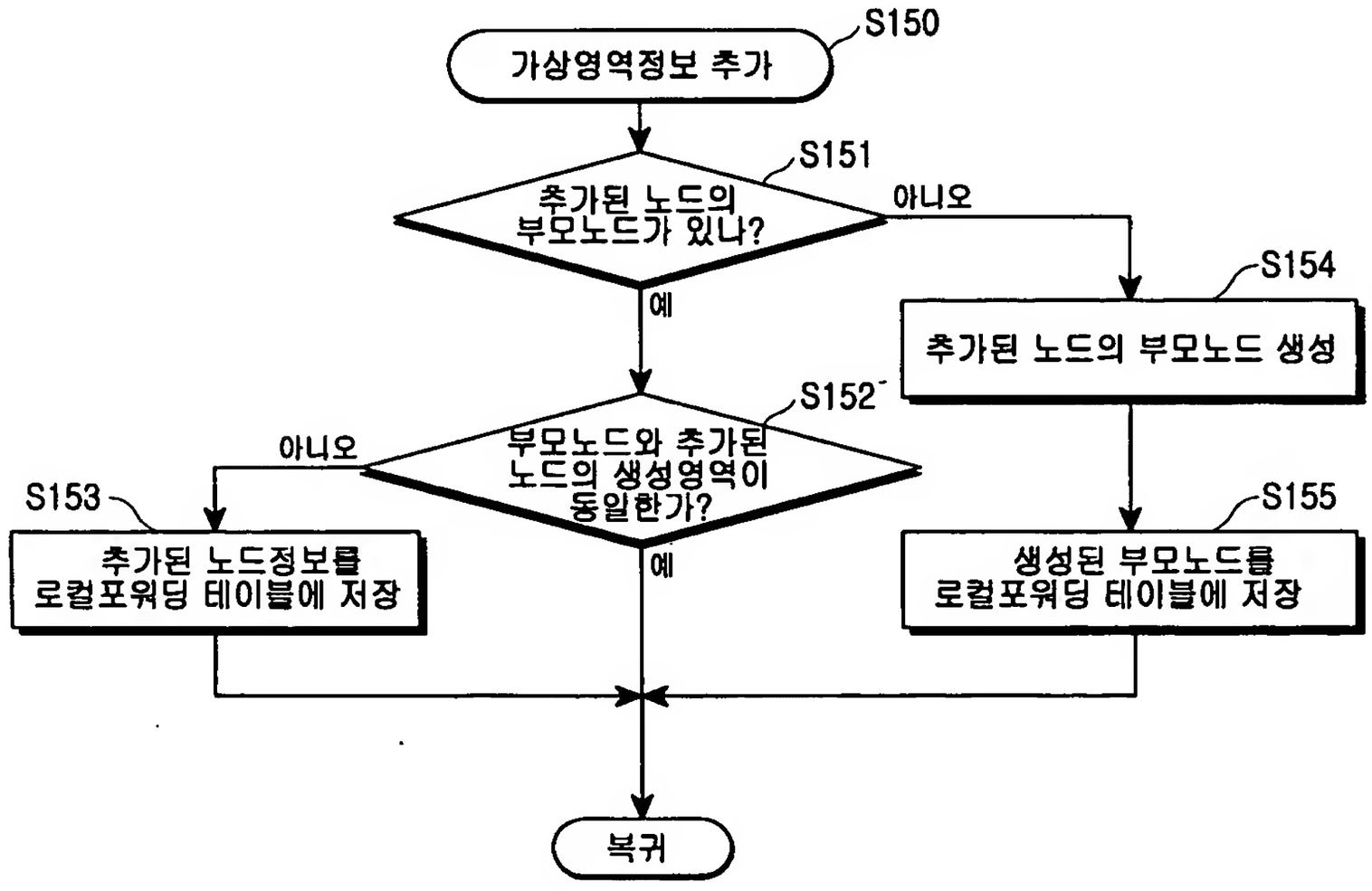
【도 5】



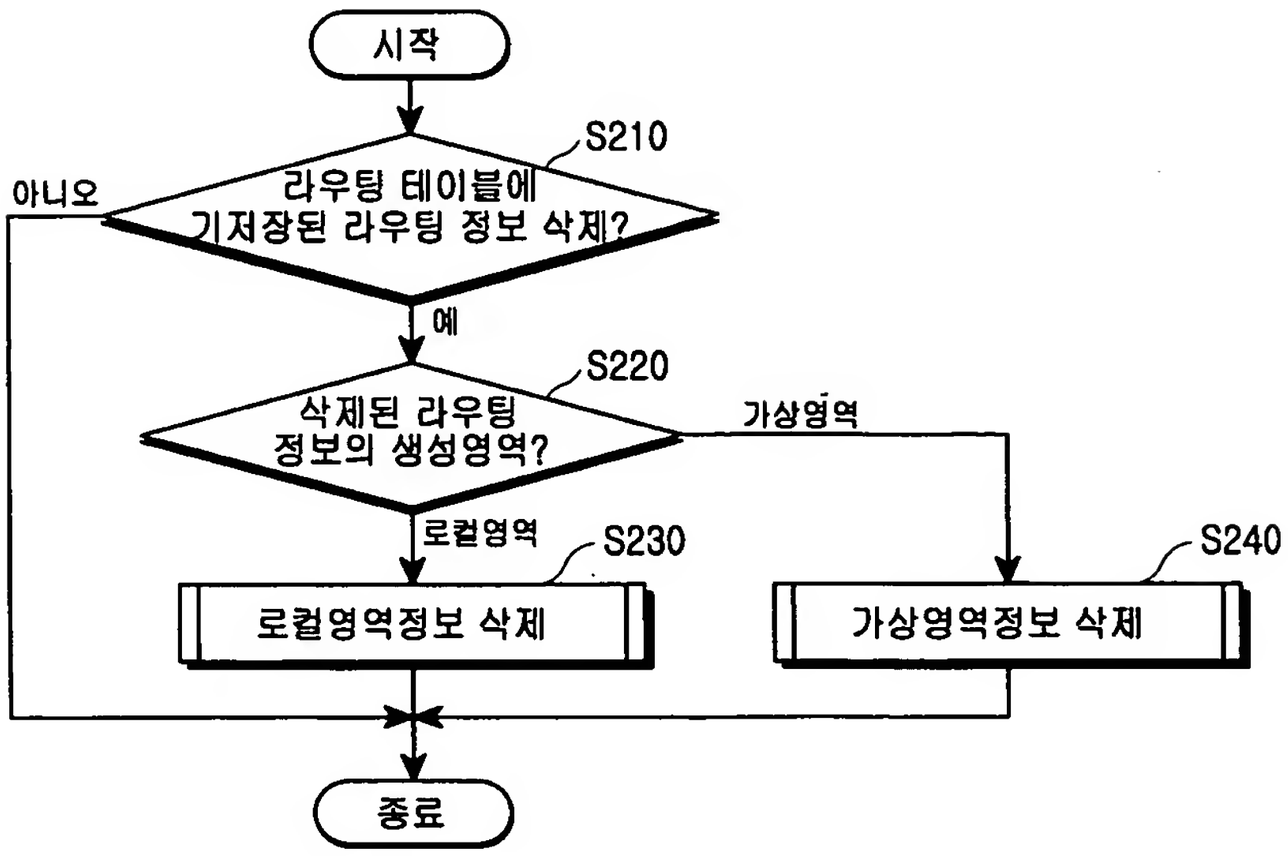
【도 6】


S140
로컬영역정보 추가
S141
추가된 노드의 부모노드가 있나?
아니오
예
S144
추가된 노드의 부모노드 생성
S142
부모노드와 추가된 노드의 생성영역이 동일한가?
아니오
예
S145
추가된 노드정보를 가상영역으로 통지(advertise)
S146
추가된 노드정보를 로컬포워딩 테이블에 저장
S143
추가된 노드정보를 로컬포워딩 테이블에만 저장
복귀

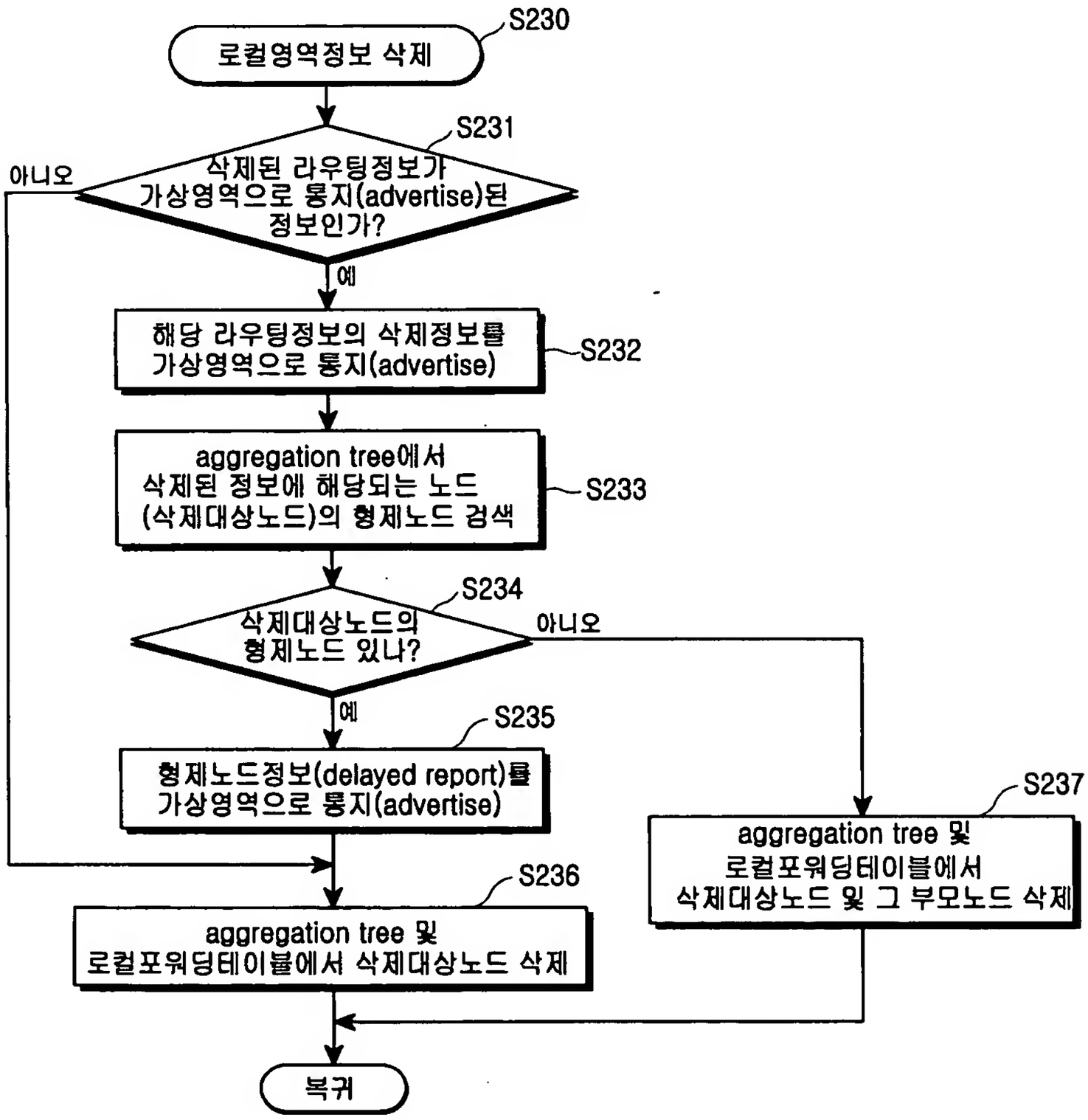
【도 7】


S150
가상영역정보 추가
S151
추가된 노드의 부모노드가 있나?
아니오
예
S154
추가된 노드의 부모노드 생성
S152
부모노드와 추가된 노드의 생성영역이 동일한가?
아니오
예
S153
추가된 노드정보를 로컬포워딩 테이블에 저장
S155
생성된 부모노드를 로컬포워딩 테이블에 저장
복귀

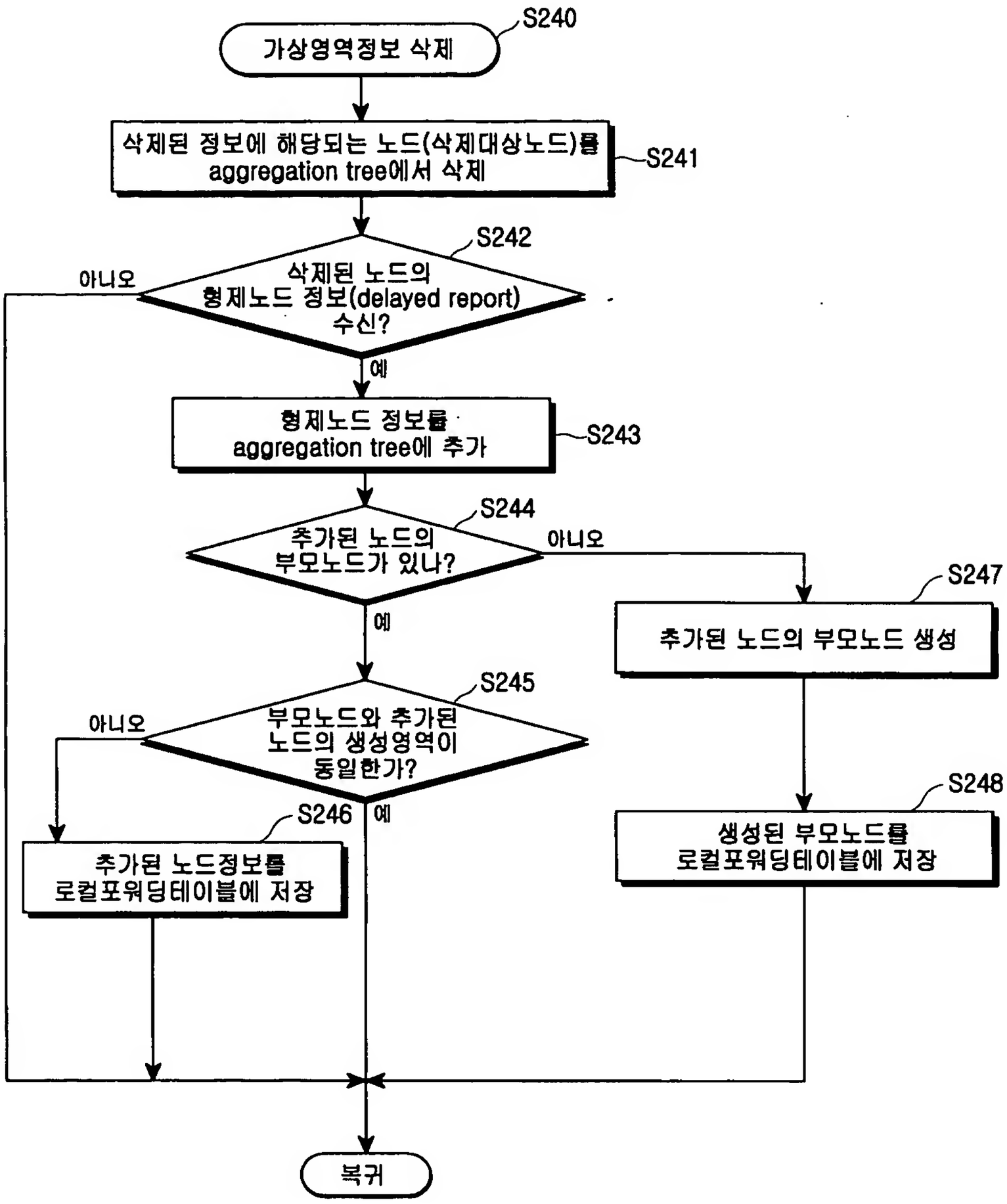
【도 8】


시작
S210
라우팅 테이블에
기저장된 라우팅 정보 삭제?
아니오
예
S220
삭제된 라우팅
정보의 생성영역?
가상영역
로컬영역
S230
로컬영역정보 삭제
S240
가상영역정보 삭제
종료

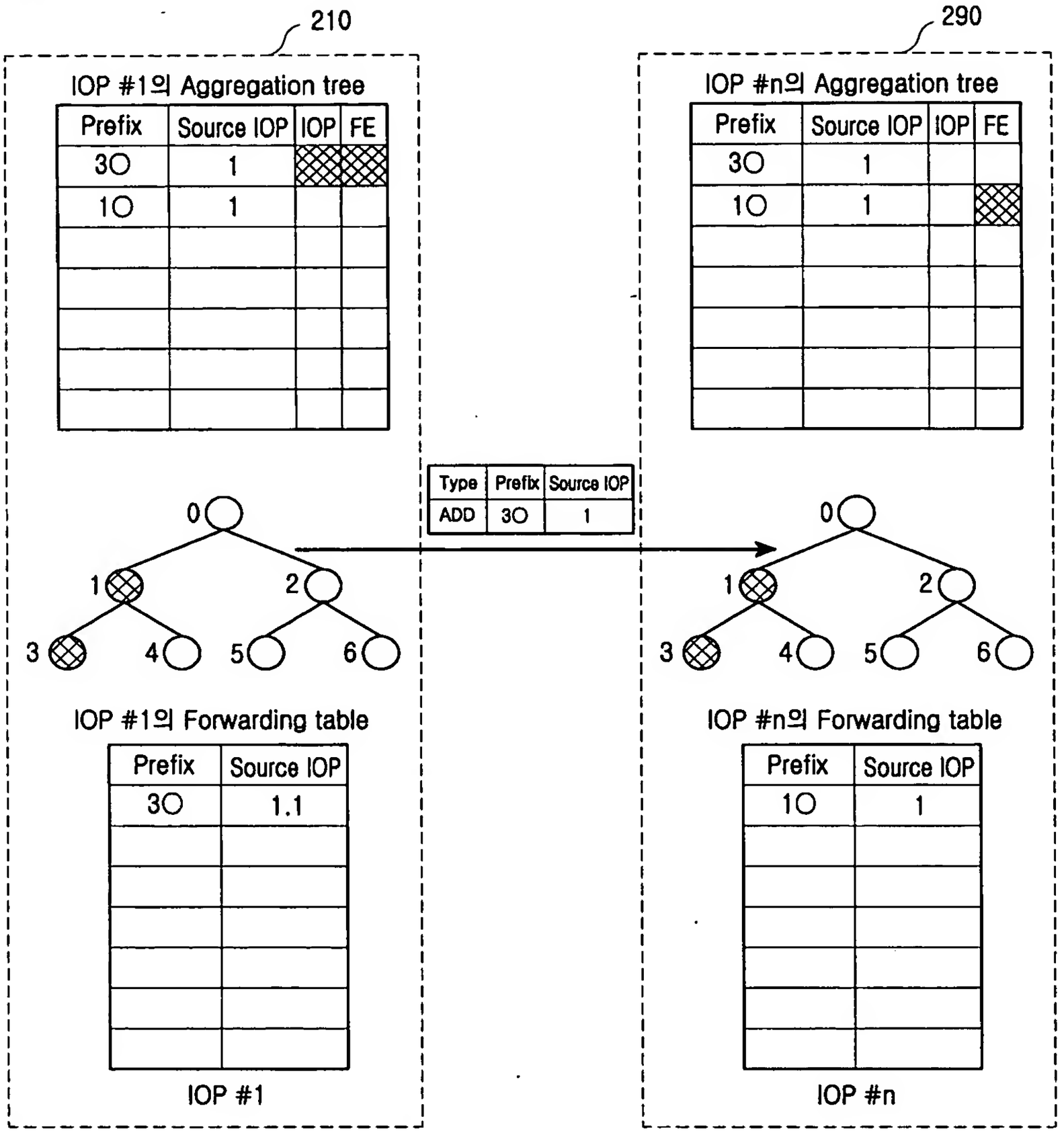
【도 9】


S230
로컬영역정보 삭제
S231
삭제된 라우팅정보가
가상영역으로 통지(advertise)된
정보인가?
아니오
예
해당 라우팅정보의 삭제정보를
가상영역으로 통지(advertise)
S232
aggregation tree에서
삭제된 정보에 해당되는 노드
(삭제대상노드)의 형제노드 검색
S233
S234
삭제대상노드의
형제노드 있나?
아니오
예
S235
형제노드정보(delayed report)를
가상영역으로 통지(advertise)
S237
aggregation tree 및
로컬포워딩테이블에서
삭제대상노드 및 그 부모노드 삭제
S236
aggregation tree 및
로컬포워딩테이블에서 삭제대상노드 삭제
복귀

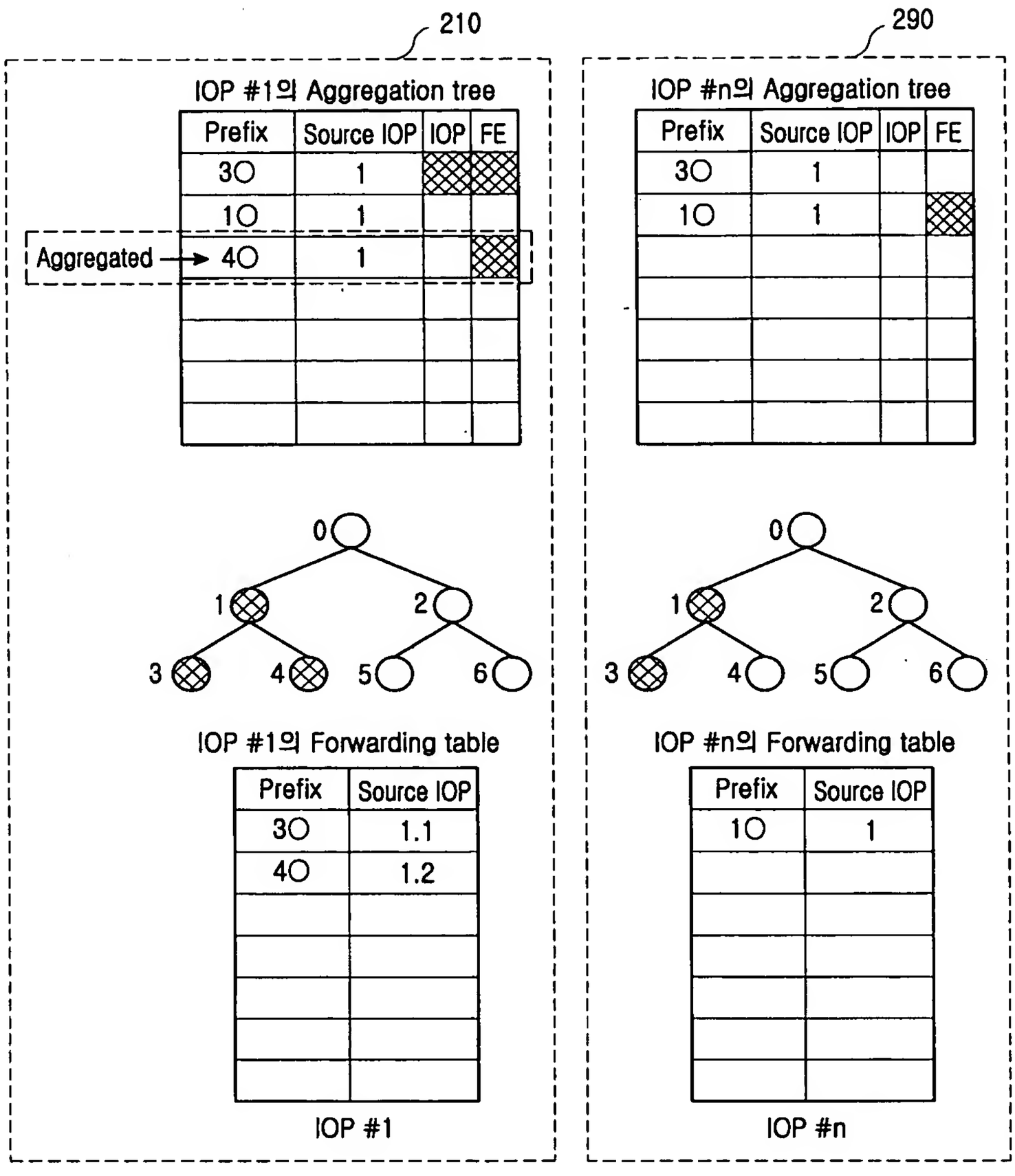
【도 10】


S240
가상영역정보 삭제
삭제된 정보에 해당되는 노드(삭제대상노드)를 aggregation tree에서 삭제
S241
S242
삭제된 노드의 형제노드 정보(delayed report) 수신?
아니오
예
형제노드 정보를 aggregation tree에 추가
S243
S244
추가된 노드의 부모노드가 있나?
아니오
예
S247
추가된 노드의 부모노드 생성
S245
부모노드와 추가된 노드의 생성영역이 동일한가?
아니오
예
S246
추가된 노드정보를 로컬포워딩테이블에 저장
S248
생성된 부모노드를 로컬포워딩테이블에 저장
복귀

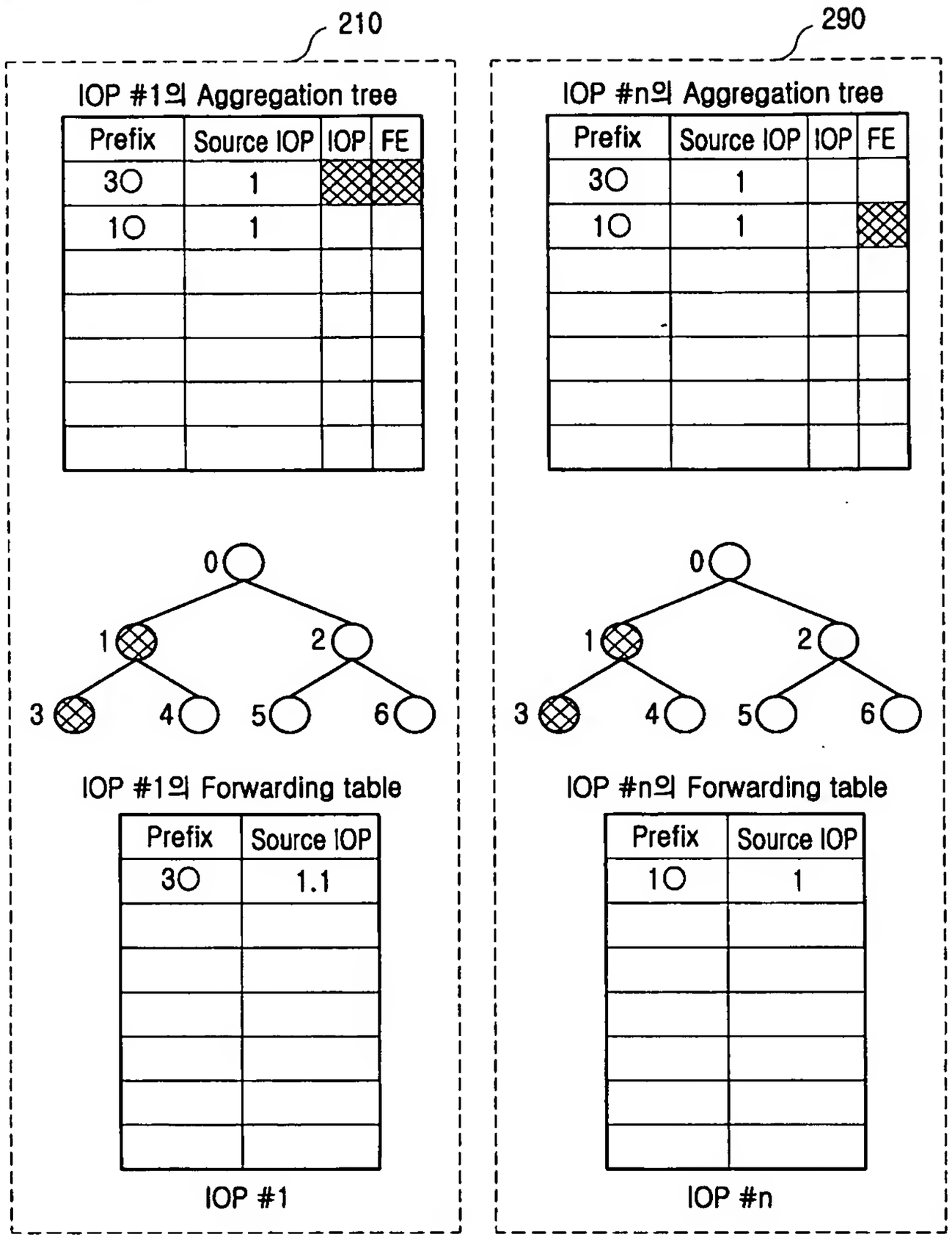
【도 11a】


210
IOP #1의 Aggregation tree
Prefix
Source IOP
IOP
FE
3O
1
1O
1
290
IOP #n의 Aggregation tree
Prefix
Source IOP
IOP
FE
3O
1
1O
1
Type
Prefix
Source IOP
ADD
3O
1
0
1
2
3
4
5
6
IOP #1의 Forwarding table
Prefix
Source IOP
3O
1.1
IOP #1
IOP #n의 Forwarding table
Prefix
Source IOP
1O
1
IOP #n

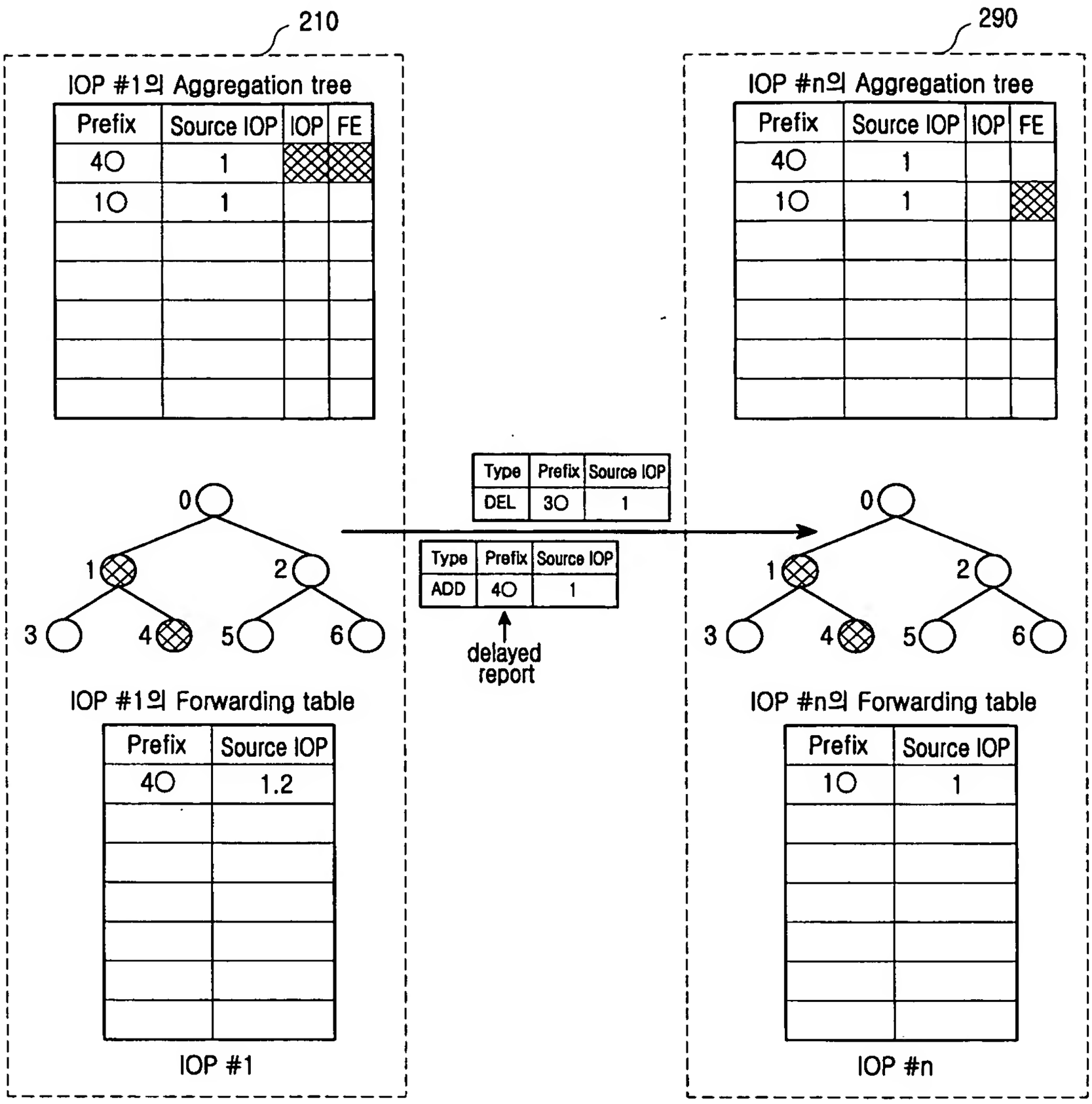
【도 11b】


210
290
IOP #1의 Aggregation tree
Prefix
Source IOP
IOP
FE
3O
1
1O
1
Aggregated
4O
1
IOP #n의 Aggregation tree
Prefix
Source IOP
IOP
FE
3O
1
1O
1
0
1
2
3
4
5
6
IOP #1의 Forwarding table
Prefix
Source IOP
3O
1.1
4O
1.2
IOP #1
IOP #n의 Forwarding table
Prefix
Source IOP
1O
1
IOP #n

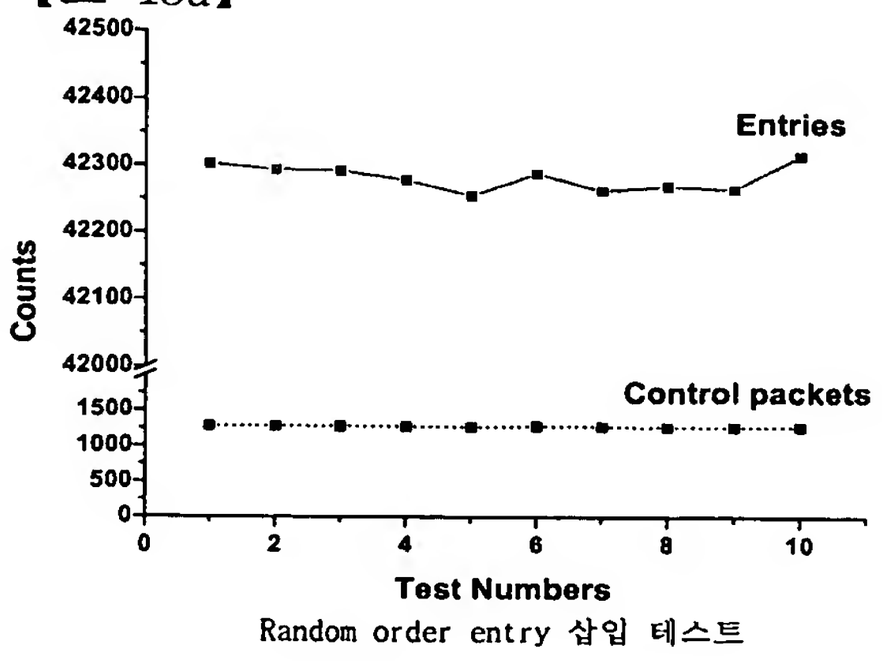
【도 12a】


210
IOP #1의 Aggregation tree
Prefix
Source IOP
IOP
FE
3O
1
1O
1
0
1
2
3
4
5
6
IOP #1의 Forwarding table
Prefix
Source IOP
3O
1.1
IOP #1
290
IOP #n의 Aggregation tree
Prefix
Source IOP
IOP
FE
3O
1
1O
1
0
1
2
3
4
5
6
IOP #n의 Forwarding table
Prefix
Source IOP
1O
1
IOP #n

【도 12b】

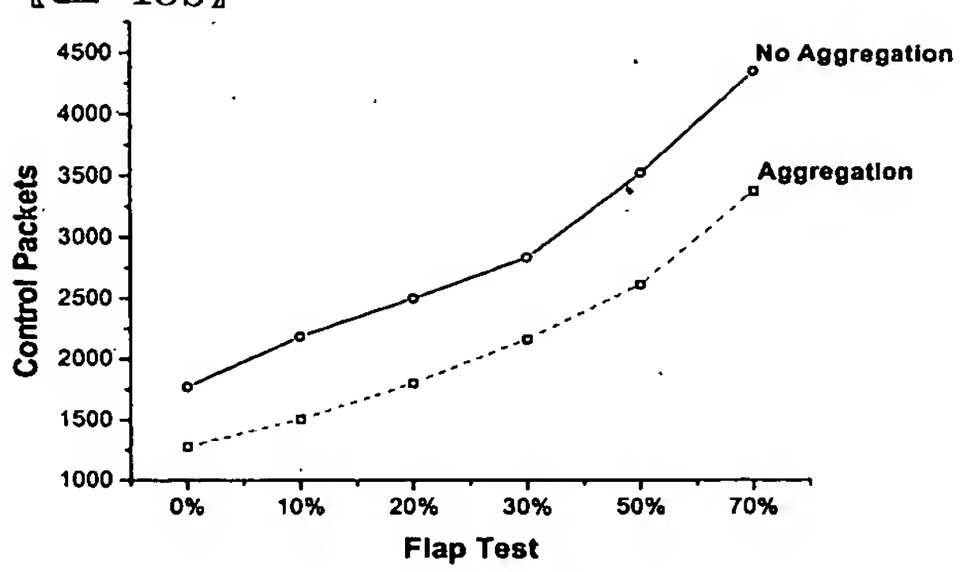


【도 13a】



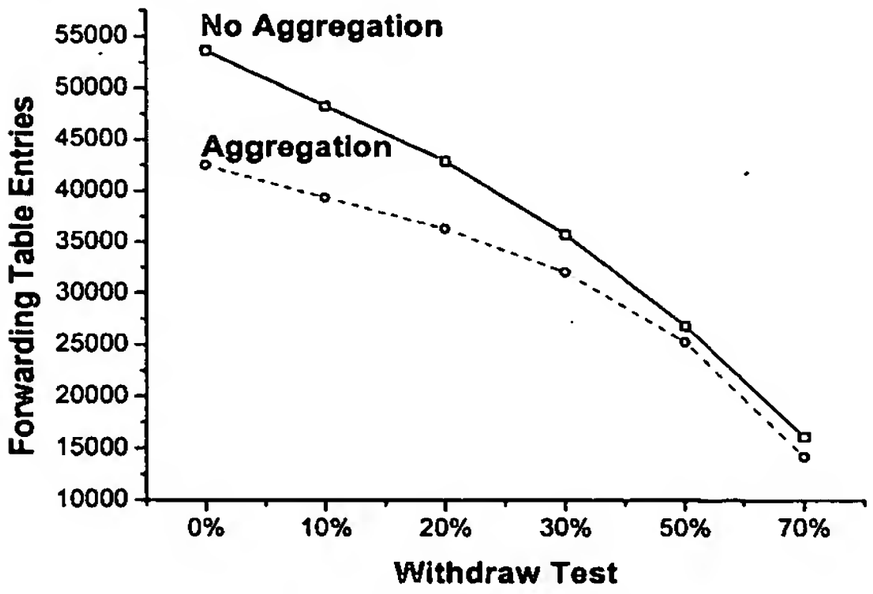
Random order entry 삽입 테스트

【도 13b】



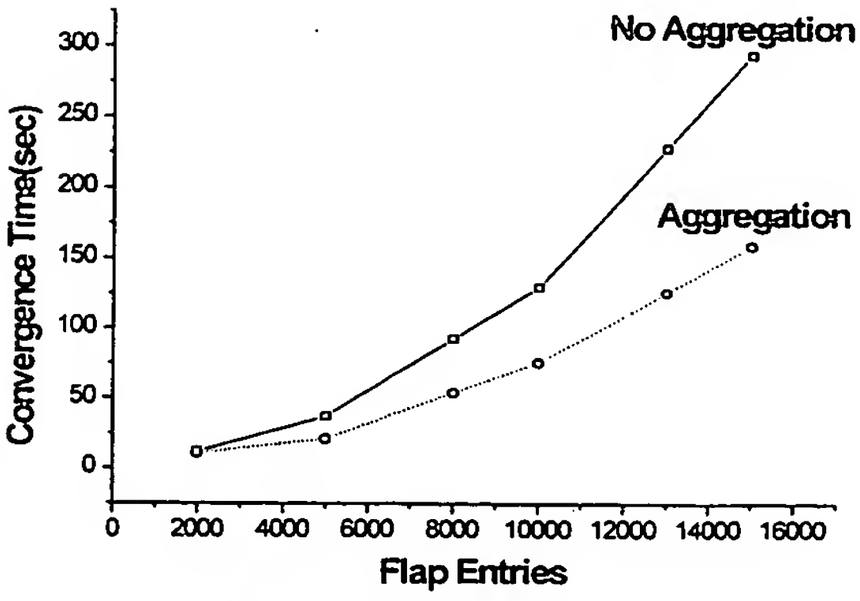
Control packet 전송량 비교

【도 13c】



포워딩 테이블 엔트리 개수 비교

【도 13d】



BGP4 flap convergence time test

【도 14a】

```
Insertion (prefix) {
 local insertionnode
 /* STEP1 */
 insertionnode := FindNode(prefix)
 /* STEP2 */
 /* Check new route's nexthop */
 if (NodeNexthopVirtual(insertionnode) = TRUE) then
    InsertionNodeVirtual(insertionnode)
 else
    InsertionNodeInter-domain(insertionnode)
 /* STEP3 */
 if (run_step3 = TRUE) then
   ChildNodeHandler(insertionnode)
}
```

【도 14b】

```
Deletion (prefix) {
 local deletenode, siblingnode
 /* STEP1 */
 deletenode := FindNode(prefix)
 /* STEP2 */
 if (DRNForward(deletenode) = TRUE) then
   if (ForwardingTableForward(deletenode) = TRUE)
     then
       DeleteForwardingTable(deletenode)
       SendDRN(deletenode)
     else
       /* Deletion of the parent node instead of  */
       /* the deletenode                          */
       DeleteForwardingTable(GetParent(deletenode))
 else
    /* In case of the deletion of the aggregation node, */
    /* sending information to DRN can be suppressed */
   if (ForwardingTableForward(deletenode) = TRUE)
     then
      DeleteForwardingTable(deletenode)

 siblingnode := SiblingNodeCheck(deletenode)
 / * Delayed insertion of the sibling node */
 if (siblingnode ≠ NIL) then
     SendDRN(siblingnode)
 /* STEP3 */
 ChildNodeHandler (deletenode)
}
```

【도 14c】

```
FindNode (prefix) {
  local node
  /* Search for node to be inserted */
  node := GetNode (prefix)
  /* Identity the insertion node */
  if ((node ? NIL) and (NodeType(node) = AGG)) then
        Empty (node)
        run_step3 := TRUE
  else
        NewNode(node)
  return (node)
}
```

【도 14d】

```
MakeParentNode (node) {
 local parentnode
 /* Make parent node and set node type to AGG */
 parentnode := AllocateParentNode(node)
 NodeType(parentnode) := AGG
 Parent(node) := parentnode
 return(parentnode)
}
```

【도 14e】

```
InsertNodeVirtul (node) {
 local parentnode
 parentnode := GetParent(node)
 if (parentnode ? NIL) then
  if (NodeSource(parentnode) ? NodeSource(node)) then
        InsertForwardingTable(node)
  /* There is not parent node. */
 else
  /* Make and wirte parent node to the forwarding table */
     parentnode := MakeParentNode(node)
     InsertForwardingTable(parentnode)
}
```

【도 14f】

```
InsertNodeInter-domain (node) {
 local parentnode
 parentnode := GetParent(node)
 if (parentnode ? NIL) then
     /* If new route source and parent are same, */
     /* new route sending to DRN can be suppressed */
   if (NodeSource(parentnode) ? NodeSource(node)) then
        SendDRN(node)
    InsertForwardingTable(node)
  else
     /* Make parent node and wirte new node */
/* to the forwarding table */
     parentnode := MakeParentNode(node)
     InsertForwardingTable(node)
     SendDRN(node)
 }
```

【도 14g】

```
ForwardingTableForwardCheck(node)
{
  local parentnode
  parentnode := GetParent(node)
  if ((ForwardingTableForward(node) = FALSE)
  and (NodeSource(node) != NodeSource(parentnode))) then
        return (FALSE)
  else
      return (TRUE)
}
```

【도 14h】

```
DRNForwardCheck(node)
{
  local parentnode
  parentnode := GetParent(node)
  if ((DRNForward(node) = FALSE)
     and (NodeNexthopVirtual(node) = FALSE)) then
      return (FALSE)
  else
      return (TRUE)
}
```

【도 14i】

```
ChildNodeHandler (node) {
 local leftchild, rightchild
 leftchild := GetLeftChildNode(node)
 rightchild := GetRightChildNode(node)
 if (leftchild ? NIL) then
    /* Disaggregation of the leftchild node */
  if ((ForwardingTableForwardCheck(leftchild) = FALSE)  then
          InsertForwardingTable(leftchild)
     /* Delayed insertion of the leftchild node */
     if ((DRNForwardCheck(leftchild) = FALSE) then
          SendDRN(leftchild)
  if (rightchild ? NIL) then
     /* Disaggregation of the rightchild node */
   if ((ForwardingTableForwardCheck(rightchild) = FALSE) then
          InsertForwardingTable(rightchild)
      /* Delayed insertion of the rightchild node */
     if ((DRNForwardCheck(rightchild) = FALSE) then
          SendDRN(rightchild)
}
```

【도 14j】

```
SiblingNodeCheck (node) {
  local siblingnode
  siblingnode := GetSiblingNode(node)
  /* Check if there is aggregated sibling node */
  if ((NodeSource(node) = NodeSource(siblingnode))
       and (DRNForward(siblingnode) = FALSE)) then
        return (siblingnode)
  else
        return (FALSE)
}
```